NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

FEATURE SETS FOR SCREENSHOT DETECTION

by

Lauren Sharpe

June 2013

Thesis Co-Advisors: Joel Young
Mathias Kölsch

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704–0188*

1. REPORT DATE *(DD–MM–YYYY)* 25–6–2013	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* 2011-06-27—2013-06-21

4. TITLE AND SUBTITLE FEATURE SETS FOR SCREENSHOT DETECTION	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Lauren Sharpe	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Department of the Navy	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number N/A.

14. ABSTRACT

As digital media capacity continues to increase and the cost continues to decrease, digital forensic examiners need progressively more efficient, effective, and tailored tools in order to perform useful media triage. This thesis documents the development of feature sets for classifying images as either screenshots or non-screenshots. Using linear- and intensity-based image information we developed the first (to our knowledge) screenshot detection algorithm. Four feature sets were developed and combinations of these feature sets were tested, with the best results achieving an F-score of 0.98 in ten-fold cross-validation. Requiring less than 0.18 seconds to analyze and classify an image, this is a critical contribution to the state-of-the-art of media forensics.

15. SUBJECT TERMS

Screenshot Detection, Computer Forensics, Triage, Feature Selection, Machine Learning, Bayes, Hough Transform, Entropy

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
Unclassified	Unclassified	Unclassified	UU	69	

THIS PAGE INTENTIONALLY LEFT BLANK

FEATURE SETS FOR SCREENSHOT DETECTION

Lauren Sharpe
Civilian, Department of the Navy
B.S., University of Virginia, 2011

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 2013

Author: Lauren Sharpe

Approved by: Joel Young
Thesis Co-Advisor

Mathias Kölsch
Thesis Co-Advisor

Peter Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

As digital media capacity continues to increase and the cost continues to decrease, digital forensic examiners need progressively more efficient, effective, and tailored tools in order to perform useful media triage. This thesis documents the development of feature sets for classifying images as either screenshots or non-screenshots. Using linear- and intensity-based image information we developed the first (to our knowledge) screenshot detection algorithm. Four feature sets were developed and combinations of these feature sets were tested, with the best results achieving an F-score of 0.98 in ten-fold cross-validation. Requiring less than 0.18 seconds to analyze and classify an image, this is a critical contribution to the state-of-the-art of media forensics.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgements

I would like to sincerely thank a number of people who helped me achieve the completion of my thesis. I have the greatest appreciation for my thesis advisors. Joel Young provided me with incredible knowledge, insight, opportunities and friendship throughout my time here at Naval Postgraduate School and during the writing of my thesis. Mathias Kölsch always guided me in the right direction when I was lost within my code or writing, and he was a voice of reason and reassurance.

Thanks to Cynthia Irvine for her commitment to the Scholarship for Service (SFS) program here at NPS. Finally, I would like to thank my friends and family for simultaneously supporting my studies and tolerating my sleep-deprivation.

Partial support for this work was provided by the National Science Foundation's CyberCorps®: Scholarship for Service (SFS) program under Award No. 0912048. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Computer forensic examiners work to find pockets of information in large quantities of binary data and tools to triage (or prioritize) this information are invaluable to the examiner. Furthermore, it can be difficult for triage tools to automatically generate meaningful descriptions of images. One useful image type is the computer screenshot.

There are a number of reasons why a screenshot may exist on a hard drive. A user may intentionally capture his screen or a portion thereof, another process running on the machine may capture and store screenshots without the knowledge of the user, or the user may have user manuals or tutorial documents containing screenshots of computer programs as part of their instructions. Additionally, web browsers such as Firefox capture and cache screenshots of user activity [1] while some malware screenshots user activity and uploads the screenshots to servers [2]. Among other purposes, screenshots can be an information source about a user's activity and can also indicate a user's involvement in the production or use of software training documents.

1.1 Research Questions

In this thesis, we present an algorithm to classify screenshot images vs. non-screenshots employing established machine learning techniques. We focused on the selection of features that resulted in the best performance of the machine learning classification algorithms, as judged by the metrics of accuracy, precision, recall and F-score.

We hypothesized that the features that best distinguish screenshots from non-screenshots are those that capture information about lines found in the image and the intensity distribution of the pixels in the image. We expected images generated from computer displays would be more likely to contain a greater percentage of long horizontal or vertical lines as compared to a non-computer-generated digital images. We also observed that an image that is not computer-generated will tend to have a more continuous and even distribution of pixel intensity values while the intensity distribution of a computer screenshot will be concentrated at certain intensity values.

From these assumptions, we built sets composed of these distinguishing features. These feature sets varied from summary values like entropy to larger feature sets consisting of pixel intensity histograms representing all 256 grayscale values. Instead of only testing individual features sets, we found the combination of our feature sets that achieved the best performance when classifying screenshots.

1.2 Computer Vision

To evaluate our feature sets, we employed a number of computer vision techniques. The field of computer vision encompasses a wide range of techniques and strategies for extracting meaningful semantic information from digital images. As humans, we often discount the difficulty of seemingly simple vision tasks in our day-to-day lives. In the advent of computer vision research, researchers assumed that the task of computer vision would be incredibly easy–at least compared to intellectual tasks such as playing chess. In hindsight, however, tasks like chess seem complex because humans have been playing chess for orders of magnitude less time than they have been using their senses to perceive the surrounding world [3]. So, while chess is hard for humans and sight is not, the opposite is true for computers. Today, there are computers playing chess at the level of a grand master [4], but computer vision has advanced only to the level of toddler [5, p. 3]. In Chapter 2, we outline the advances in computer vision relevant to the computer vision task at hand. These advances touch on topics such as machine learning techniques for classification of images as well as means of generating features for use in the machine learning algorithms.

1.3 Significant Findings

Our research developed four feature sets

1. Line Segment Percentages (LSP)
2. Line Segments Binned by Length (LSB)
3. Intensity Entropy (CE)
4. Intensity-Based Histograms (CB)

and tested the sets both individually and combinatorially while also varying parameters associated with the calculations of the feature sets themselves. Our ten-fold cross-validation testing returned results with F-scores as high as 0.98 (when combining all four feature sets) and F-scores still between 0.96 and 0.98 when a combination of any three of the feature sets were used. After testing these four particular feature sets, we were able to determine the best combination of the feature sets, the marginal successes derived from each added feature set, and the best parameter choices for the classification algorithm.

1.4 Thesis Structure

The next chapter in this thesis (Chapter 2) provides an overview of the techniques used in this research as well as a background of the relevant research in this area. Chapter 3 presents the methodology used as well as a detailed description of the features extracted and machine learning algorithms employed. Chapter 4 documents the results of the experiments and presents summary statistics as well as examples of images. Finally, Chapter 5 recaps and draws conclusions about the work performed and develops a road map for future research in this area.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Background

This chapter documents the research most relevant to the task of screenshot classification. It begins with the high-level task of scene recognition and then expands upon the details of lower-level computer vision techniques applied to perform scene recognition.

2.1 Scene Recognition Related Work

The techniques of computer vision can be applied to a number of problems, from facial recognition to satellite imagery analysis. One such problem is scene recognition. Xiao et al. define a scene as a "place [within] which a human can act..., or a place to which a human could navigate" [6]. In the past few decades there has been a significant amount of work on this task using varying approaches. While scene recognition is a technique for classifying images, more specifically these images tend to be a digital representation of a physical location. According to the definition proposed by Xiao et al., our work—screenshot detection—is more accurately described as classifying non-"scenes" from "scenes" as opposed to the typical scene recognition task of differentiating one scene category from another.

Scene recognition usually consists of two main components: feature selection and machine learning. Existing scene recognition research encompasses a variety of features which can be grouped into related categories. A popular feature category for scene recognition (among other machine learning tasks) is codewords. Codewords are local invariant descriptors that are gathered via some means, recorded (often in a dictionary), and applied to recognition or classification tasks. Textons, scale-invariant feature transform (SIFT) and speeded up robust features (SURF) are three local feature detectors that are commonly used in recognition tasks.

Stanford's Fei-Fei Li has done a great amount of work on scene recognition. In a 2005 paper, she presented a method of unsupervised learning with Bayesian classification to categorize image scenes using codewords as features [7]. Just a year earlier, David Lowe developed a method for extracting features from images that is invariant to scale and rotation, with the goal of matching objects between images with different views of an object or scene [8]. This process, the scale-invariant feature transform (SIFT), has since been widely applied to the problem of scene recognition [9, 10]. Others approach the classification task not by identifying individual elements of an image or scene, but by classifying the overall structure of the image (at a high-level) into what authors Oliva and Torrabla termed “Spatial Envelopes” [11].

Lowe, in his 2004 paper introducing SIFT features, notes that the success of his proposed features lies in their ability to “identify objects among clutter and occlusion” with the intention of identifying the same object from two differing viewpoints [8, p.1]. Bay et al. improved upon SIFT feature by developing SURFs, which outperformed SIFT features in a number of areas including repeatability and speed while maintaining their robustness [12]. For the classification task of this thesis, we examine two distinct categories of images, only one of which is a “scene” in the traditional sense. The concept of visual viewpoints (as mentioned in the SIFT paper), while intuitive regarding scene-based images, becomes difficult to define for computer screenshots. Furthermore, computer screenshots are usually not cluttered with objects in the same way that photographs are. Codeword-based feature sets attempt to select distinctive local feature points which can easily be matched. Since screenshots are likely to have regions of uniform pixels, codeword detectors would not be able to find distinctive local pixel groups within these areas.

Color-based features are also useful in computer vision tasks. In 1991, Swain and Ballard introduced the visual cue of an object’s color-histogram to assist a robot in identifying objects [13]. One problem with using a histogram of pixel values is that if an object is a slightly different shade of a color it would have different pixel values. Increasingly, scene classification researchers are developing algorithms that combine multiple feature sets (sometimes both high- and low-level) features in order to improve their classification success [14].

Due to the concerns described above, and the novelty of the goal of detecting screenshots from non screenshots (as opposed to the more generic goal of distinguishing between classes of scenes), we chose to develop our own tailored feature sets. Our algorithm uses a number of feature sets that it tests both individually and in combination. The computer vision fundamentals required to understand these feature sets (which will be described in detail in Chapter 3) are documented in the remainder of this chapter.

2.2 Feature Selection

An important step in any classification task like scene recognition is feature selection. Two categories of features that seem promising in the classification of screenshots are linear-based features and intensity-based features. A screenshot should have significantly more horizontal and vertical lines than a typical photograph or non-computer-generated image. Also, it is likely that a computer-generated image such as a screenshot will contain large sections of pixels with identical intensities as well as a less "natural" pixel distribution overall.

2.2.1 Edge Detection

Two basic and related linear features present in digital images are edges and lines. An edge is defined as a significant change in pixel intensity and can be detected using first - and second-order derivatives of the gradients of pixel intensity [15, p. 541]. In 1963, Lawrence Roberts authored a paper titled "Machine Perception of Three-Dimensional Solids" in which he proposed what would become one of the first edge detection algorithms [16]. His algorithm applied two 2x2 masks as convolution filters to an image. One filter detected horizontal changes in intensity, and the other vertical. Another method, presented by Sobel, involves 3x3 masks that calculate both the magnitude and direction of the egdes [15]. A number of additional edge detectors have been developed, including Prewitt, LoG, and Canny [15, 17, 18].

2.2.2 Line Segment Detection

While edge detection algorithms operate on small, local regions of an image, in order to extract more meaningful content we can apply an algorithm that performs line detection. Lines, as defined in the 1986 paper "Extracting Straight Lines" are a "collection of local edges that are contiguous in the image" [19]. Burns et al. go on to assign lines attributes such as length and width. Two decades before Burns, Paul Hough developed a straightforward and successful method for global pattern recognition which he patented and which can be applied to the task of line detection [20]. Hough line detection begins with the collection of edge points from an image and employs a method of voting on the parameter space of lines in order to determine the lines that best fit the collection of edge points. This method requires some tuning to find the optimal number of lines for a given image. After line detection, the Hough algorithm uses the parameters of the lines as well as the end pixels in order to reduce the lines (which are of infinite length) to line segments (which have finite length and specified endpoints) [20].

Burns and Hough's definition of a line differs slightly from that of Haralick and Fischler who describe a line as a region whose pixel intensities differ significantly on both sides of the region of interest (as opposed to just one side) [21, 22]. Since the 1980s, researchers have made an attempt to improve upon traditional line detection algorithms with respect to their speed, accuracy, and false detection control [23, 24], but the Hough transform method has remained a reliable and accurate technique. For our feature extracting process we used a probabilistic Hough transform as implemented in the OpenCV library for Python [25].

2.2.3 Intensity-based Features

As mentioned earlier, two promising intensity-based features for screenshot detection are contiguous sections of the same pixel color and the distribution of the entire image's pixel colors. One method of determining the first of these two features is run-length encoding. Run-length encoding is a method of representing the length of stretches of the same color pixel in an image. Long runs of the same pixel should be more likely to occur in a computer-generated image than in a digital photograph.

While run-length encoding only addresses contiguous colors in one dimension, the algorithm could be modified to measure two-dimensional patches of color as well. Fisher and D'Amato combined run-length encoding and the Hough transform for detecting document skew [26]. In the same vein as run-length encoding, it is likely that the histograms of screenshots and of photographs would be noticeably different due to the lower entropy of colors displayed on a typical user's computer screen at a given time as compared the wide variety of colors in a photograph. For the purposes of our algorithm, and due to the nature of our data sets (which will be described in detail in Chapter 3), we converted all images to grayscale during image pre-processing and employed intensity-based features as opposed to color-based features. Our method implemented a histogram-based feature set for pixel intensities as opposed to run-length encoding in order to better capture information about pixels with the same intensities that are discontinuously located. Chapter 5, however, discusses the use of run-length encoding as future work.

2.3 Machine Learning

An important part of an automated classification algorithm are the machine learning techniques employed to perform the classification. Because the main focus of this research is on the feature set selection, we elected to only consider some simpler (yet still effective) techniques for the machine learning component of the classification algorithm.

2.3.1 Naïve Bayes

Bayesian learning provides a probabilistic means of classifying an entity based on training from previously-observed and categorized data points. The simplest form of Bayesian learning is the naïve Bayes classifier which significantly reduces the number of probabilities needed by assuming that all features in the classifier are conditionally independent given the class [27]. While this assumption is generally incorrect, for many application domains naïve Bayes classifiers have performed comparably to more complex machine learning techniques such as decision trees and neural networks [28]. Bayesian classifiers are a routine method of image scene classification [7,29].

2.3.2 Support Vector Machines

Another machine learning technique is the support vector machine(SVM). Introduced by Vapnik and Cortes in 1995, SVMs work by mapping data points into high-dimensional spaces in order to find a hyperplane that can separate data points from different classes [30]. SVMs combine multiple learning methods like linear learning machines and kernel-based feature transformations in order to be an effective and computationally efficient machine learning technique [31]. Researchers have used SVMs as a means of image classification, including histogram-based classifications [32,33].

In order to best analyze the results of the feature selection we decided to choose one machine learning algorithm to use consistently for the duration of the experimentation and testing. We used a Bayesian classifier due to its simplicity and relative computational efficiency, but future work could include testing feature sets across multiple machine learning algorithms.

The remaining chapters document the methodology involved in applying the aforementioned techniques to perform image classification. The developed algorithm will employ linear and intensity-based image attributes as features in machine learning mechanisms in order to detect computer screenshots from a set of digital images.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Experiment Design and Implementation

In this chapter, we discuss our application of the machine learning and feature extraction techniques introduced in the previous chapter. We present our proposed methodology for screenshot detection and the framework within which we conducted the testing of our algorithms.

3.1 Features

As stated earlier, the main focus of this thesis was to determine a combination of features that were well-suited for the classification task at hand. We looked at two main categories of feature sets: those representing the linear features of an image and those representing the intensity-based features. For each of these categories we identified one set of features that is more cumulative in nature and one that provides more detail. While the feature sets are first presented individually, our experimentation examined all combinations of feature sets in order to identify the combination of features that yielded that best performance of our algorithm.

3.1.1 Feature Set 1: Line Segment Percentages (LSP)

Our first set of features consisted of three features pertaining to line segments in an image. To generate this feature set, we first applied a Hough transform, a standard computer vision technique for line segment detection. The result of this transform was quantitative information about the line segments found in the image from which we were able to extract the orientation of each segment. We then counted the number of line segments that qualified as horizontal and the number that qualified as vertical based on Equation (3.2) and were within a margin of about 2°, or $threshold_{slope} = 30$. The three features were the percentage of line segments that were vertical, the percentage of line segments that were horizontal, and the percentage of line segments that were either horizontal or vertical.

$$slope = \frac{y_2 - y_1}{x_2 - x_1} \tag{3.1}$$

$$orientation = \begin{cases} horizontal & \text{if } |slope| < \frac{1}{threshold_{slope}} \\ vertical & \text{if } |slope| > threshold_{slope} \\ neither & \text{otherwise} \end{cases} \tag{3.2}$$

$$LSP_{either} = \frac{|segments_{horiz}| + |segments_{vert}|}{|segments_{all}|} \tag{3.3}$$

$$LSP_{horiz} = \frac{|segments_{horiz}|}{|segments_{all}|} \tag{3.4}$$

$$LSP_{vert} = \frac{|segments_{vert}|}{|segments_{all}|} \tag{3.5}$$

Where $segments_{horiz}$, $segments_{vertical}$ are the sets of line segments whose $orientation$ is $horizontal$ or $vertical$, respectively, and $segments_{all}$ is the set of line segments whose $orientation$ is $horizontal$, $vertical$, or $neither$.

3.1.2 Feature Set 2: Line Segments Binned by Length (LSB)

While the LSP feature set provided a good summary of the vertical and horizontal lines within an image, it neglected to account for the lengths of the vertical and horizontal line segments. In the LSB feature set we again obtained the extracted line segment information that we used for the LSP feature set. We then looked at only those line segments that qualified as horizontal or vertical (using the standard employed in the previous feature set and documented in Equations (3.1)–(3.5)). However, instead of aggregating this information into percentage-based features, we aggregated the information into a histogram binned by length of the line segment.

In order to account for images of differing size, we normalized the length of the line segment by the size of the image in that direction (width for horizontal line segments and height for vertical ones). This normalization provided us with the length of the line segment as a percentage of the image size rather than in pixels. During experimentation, we also varied the number of bins (and therefore bin size) in order to observe how bin size affected our classification success.

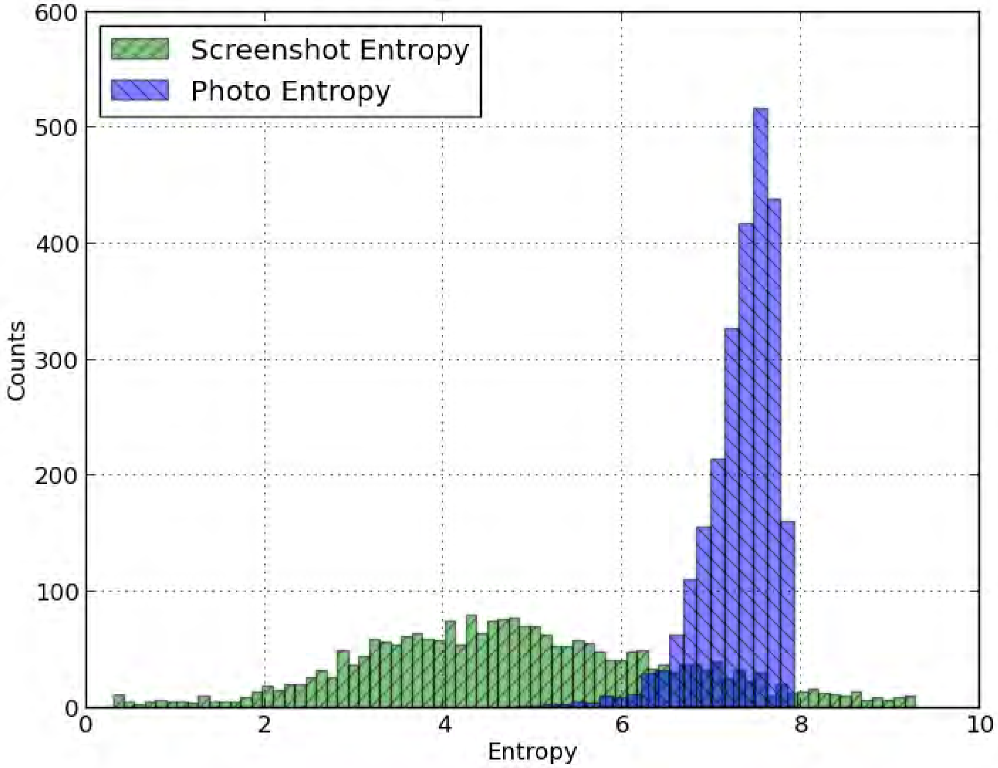


Figure 3.1: Intensity Entropies of Images by Class. While there is some overlap in the values of the entropy between classes, they are clearly distributed differently. Entropy should be a distinguishing feature and may prove useful especially in combination with other features.

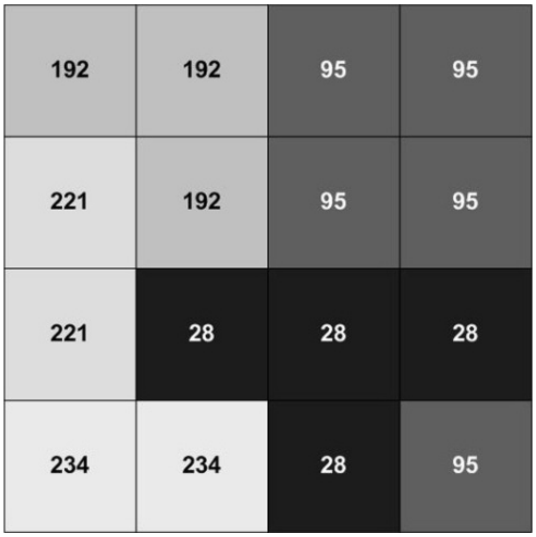
3.1.3 Feature Set 3: Intensity Entropy (CE)

For this feature set (CE), we wished to capture information about the intensity distribution of the image. First we used a single summarizing feature of the entropy of an image. In order to maintain consistency across color images and grayscale images, we converted all images to grayscale before calculating the intensity entropy of the pixel values. As evidence justifying the use of entropy as a feature, we calculated the intensity entropies of all the images in our dataset and plotted them in Figure 3.1.

3.1.4 Feature Set 4: Intensity-Based Histograms (CB)

Another promising set of features for distinguishing screenshots from non-screenshots are more robust intensity-based features. A priori, we expect that images like photographs have a more even distribution of pixel colors, while screenshots may have large spikes at various pixel values, since large portions of a computer screenshot may be the exact same color. It is possible that these spikes occur at the same pixel value across multiple screenshots, for example, many screenshots may have an usually large number of pure white pixels as compared to photographs. Other spikes may be less easily pinpointed at specific pixel values.

If it were the case that the location of spikes (which correspond to the intensity value) were consistent from screenshot to screenshot, an appropriate feature vector would likely be just a histogram of pixel intensity values. However, because of the variability of the location of spikes in the histogram, we chose to normalize the histogram of pixel values (which have been placed into 256 bins, one per intensity value) and then bin the value of these bin heights in order to generate a representation of the distribution of pixel intensities. After this second binning, we hypothesized that photographs (non-screenshots) would have high counts in lower bins indicating a fairly even distribution of pixel intensities. We also expected the screenshots to have higher counts in the higher bins than photographs (due to pixel-value spikes in the original grayscale value histogram). During our experimentation we varied the number of bins in the second histogram (in a similar way as the LSB feature set). Figure 3.2 provides a sample image from which we will step through the CB feature set calculations.



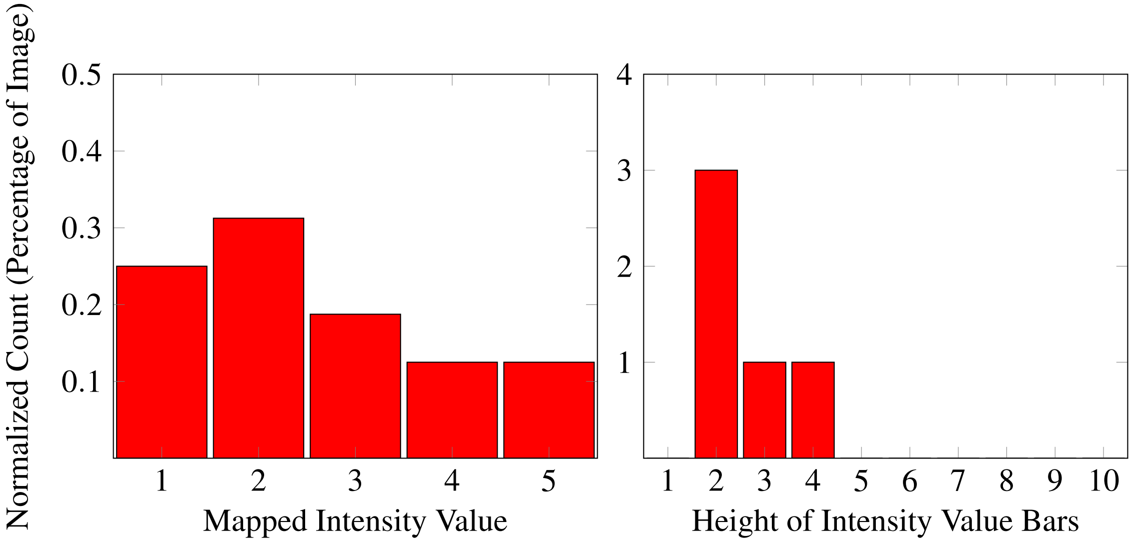
192	192	95	95
221	192	95	95
221	28	28	28
234	234	28	95

Figure 3.2: CB Feature Set Example. For simplicity's sake, in this example we will map the five distinct grayscale intensities to the values one through five.

Original Intensity Value	Mapped Value	Count	Normalized Count
28	1	4	0.25
95	2	5	0.3125
192	3	3	0.1875
221	4	2	0.125
234	5	2	0.125

Table 3.1: CB Calculations

The algorithm first generates the counts of each intensity, and then normalizes them by dividing by the total number of pixels in the image. Table 3.1 shows the results of these calculations.



(a) Histogram of Intensity Values

(b) Histogram of Normalized Count of Histogram Values

Figure 3.3: Histograms Calculated for CB Feature Set

Figure 3.3(a) provides the percentage of pixels with each intensity value in the image. For real images, there are be 256 values, one for each possible grayscale pixel value. This histogram is useful for identifying specific values that occur more frequently, in this case, value “two.” However, for our research it was more valuable to know that any value occured with a high frequency, and less important to know which specific intensity value has the high frequency. To achieve this, we built another histogram from the normalized count values. An example of this second histogram is presented in 3.3(b).

Continuing with our example, we choose a bin size of ten. We know that our values in the first histogram (Figure 3.3(a)) will not exceed 1.0 since they are percentages, so our ten bins each have a width of 0.1.

In Figure 3.3(b), histogram two shows that three of the intensity values occured with a frequency between 0.1 and 0.2, one value had a frequency between 0.2 and 0.3 and the final value had a frequency between 0.3 and 0.4. The histogram, however, no longer distinguishes which grayscale intensity values correspond to which frequencies.

3.2 Experimental Framework

Our experimental framework consisted of two datasets and a Python program with the functionality for both the feature extraction and machine learning algorithms. The python program had the capability to train and test images or to run on pre-trained classifiers in order to just test a set of images.

3.2.1 Datasets

For these experiments we needed a minimum of two datasets, one set of screenshots and one of non-screenshots. There are a number of published image datasets that have been used for a variety of computer vision experiments and papers. For our non-screenshot dataset, we wanted a generic yet diverse set of photographs and decided upon Fei-Fei Li's "13 Natural Scene Categories" dataset from her 2005 paper [7]. We were not able to find an established dataset of screenshot images, so we chose to create our own dataset of screenshots from Wikimedia Common's "Screenshots" category [34]. At the time of the corpus's retrieval (February 2013), the Screenshots category contained approximately 2,700 images that the users of Wikimedia had tagged with the label "Screenshot." After retrieving these images from Wikimedia, we then filtered the images to fit a more specific definition of a screenshot.

For our purposes, we required that a screenshot be captured by the device itself (and not, for example, a digital camera external to the system). The screenshot needed to have a definitive element of a computer screen, such as a menu bar or desktop icon. For example, a screenshot that only captured a region of a computer screen that was showing a photograph would be visually indistinguishable from the photograph itself. After removing the images that did not meet these specifications, we had a screenshot dataset containing approximately 2,400 images.

3.2.2 Experiments

In order to successfully perform these experiments, we used a number of Python libraries. Python Imaging Library [35] and OpenCV [25] provided image processing and feature extraction capabilities, NumPy [36] was used for general numeric processing and the Orange [37] data mining library provided an implementation of a Bayesian classifier.

For all of our experiments, we ran a naïve Bayes classifier from Orange (with the default parameters) on a combination of the previously mentioned feature sets. We exhaustively combined all of the feature sets for a total of 15 experiments. Orange's naïve Bayes implementation uses the LOESS locally-weighted regression [38] to estimate the conditional probability distribution.

In order to minimize our search space, we first ran isolated tests on the experiments for the LSB and CB feature sets. These feature sets involve binning and require a parameter as input for the number of bins used to create the feature vector. From these experiments, we selected the best parameter values. These two parameters were then used for the remaining 11 combinatorial experiments. The use of other machine learning algorithms as well as a structured experimental comparison to other scene classification algorithms are two topics for future work that are discussed further in Chapter 5.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Analysis of Experiments

The experiments described in the previous chapter were evaluated according to accuracy, precision, recall, and F-score and the results are presented in this chapter. We conducted 15 experiments by trying all combinations of our four different feature sets within the framework of a naïve Bayes classifier. Our classifier ran on two data sets of 2400 images each–one of screenshots (the positive class for the machine learning algorithm) and one of non-screenshots (the negative class). This chapter is structured to present detailed analysis and examples of the first four experiments (testing each feature set individually), and then summary statistics on the remaining combinatorial experiments, while bringing attention to experiments of note. Any performance metrics presented here are a result of a ten-fold cross-validation run on all 4800 images.

4.1 Feature Set 1: LSP

Our first experiment used the first feature set consisting of only three features. These three features were the percent of line segments that were horizontal, the percent of line segments that were vertical, or the percent that were either horizontal or vertical.

Table 4.1 presents the confusion matrix for the experiment containing only the LSP feature set.

	Predicted Screenshot	Predicted Non-screenshot
Actual Screenshot	2263	153
Actual Non-screenshot	125	2291

Table 4.1: LSP Feature Set Confusion Matrix

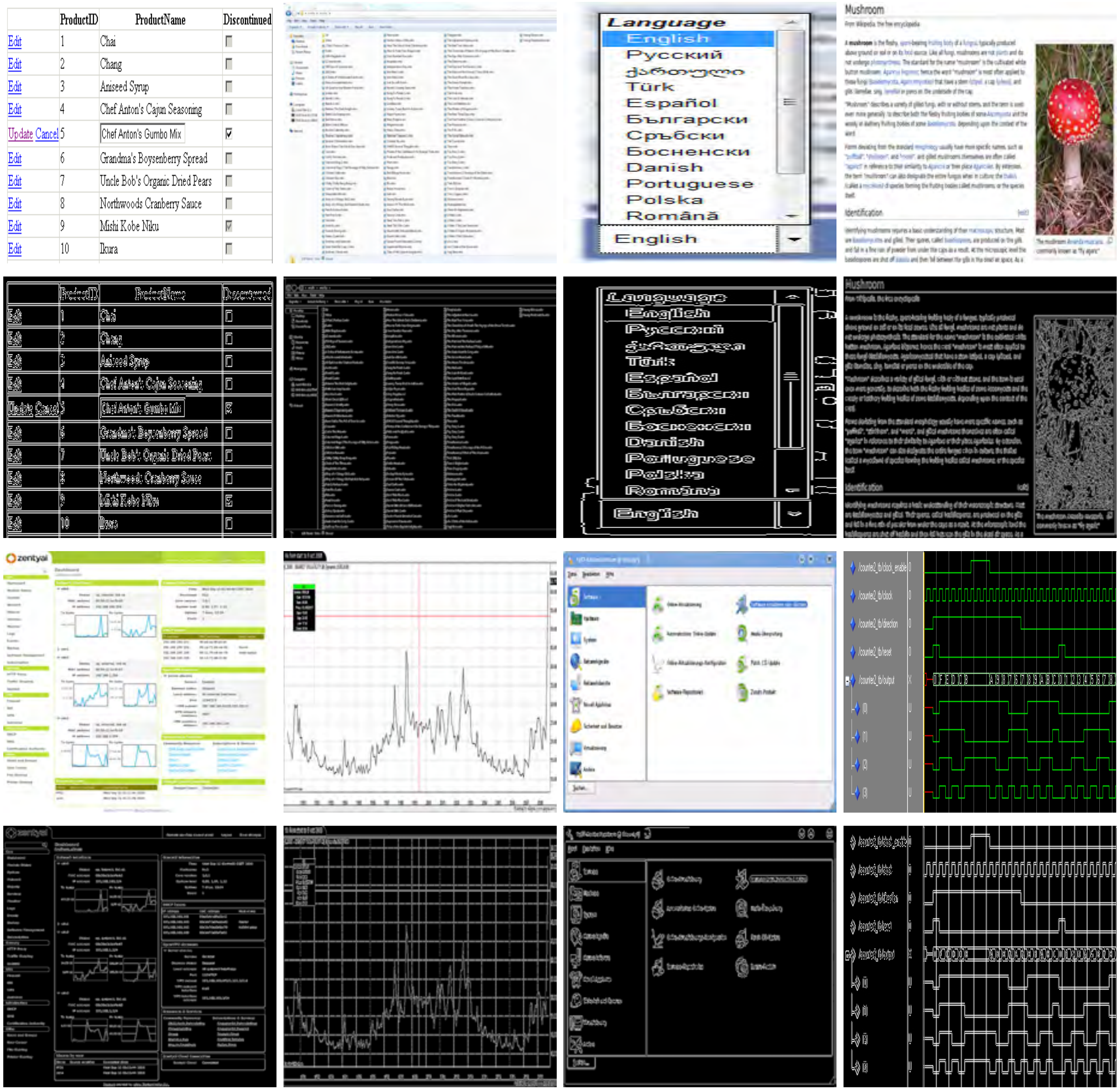
Accuracy	Precision	Recall	F-score
0.942	0.948	0.937	0.942

Table 4.2: LSP Results

Table 4.3 lists the average values of each of the three features for each class. While the horizontal percent differs greatly between the classes (with a range of 0.571), the vertical percent has a much closer margin (with a range of 0.042).

4.1.1 Example Images

In Figures 4.1–4.4 we present four sets of images from the LSP feature set experiment. The images are shown with their corresponding line segment extractions.



	ProductID	ProductName	Discontinued
Edit	1	Chai	☐
Edit	2	Chang	☐
Edit	3	Aniseed Syrup	☐
Edit	4	Chef Anton's Cajun Seasoning	☐
Update Cancel	5	Chef Anton's Gumbo Mix	☑
Edit	6	Grandma's Boysenberry Spread	☐
Edit	7	Uncle Bob's Organic Dried Pears	☐
Edit	8	Northwoods Cranberry Sauce	☐
Edit	9	Mishi Kobe Niku	☑
Edit	10	Ikura	☐

Figure 4.1: True Positives Using LSP Feature Set. This set of images is of true positives (images that the Bayesian classifier determined were screenshots and were indeed screenshots). It is evident from the accompanying line detection images that horizontal and vertical line segments are prominent in the images.

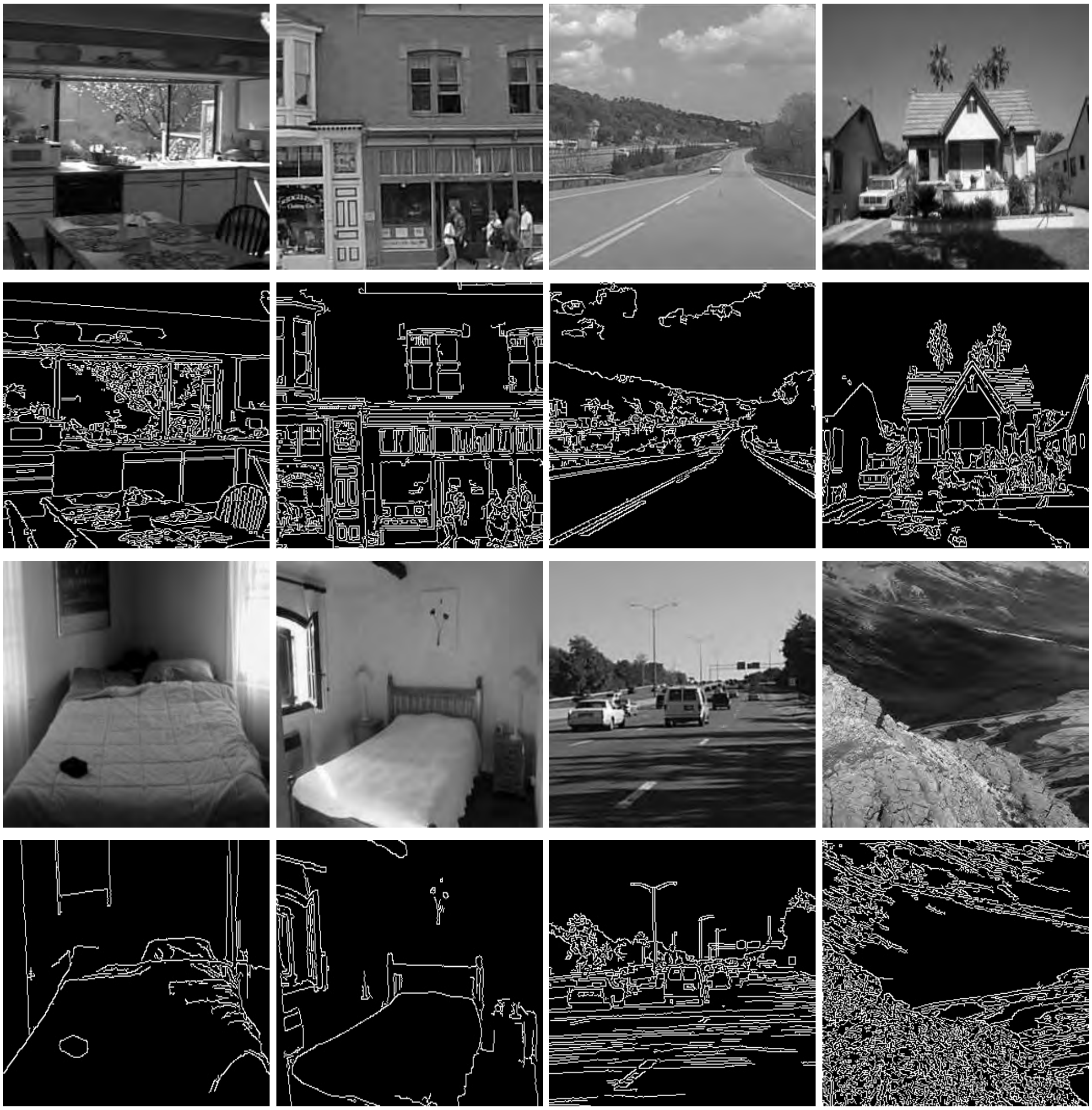


Figure 4.2: True Negatives Using LSP Feature Set. This set of images is of true negatives (correctly classified as non-screenshots). While these images may have some (or even many) line segments that are horizontal or vertical, they have a high number of line segments that are not horizontal or vertical, which significantly affects the percentage calculation and provides for an accurate classification as non-screenshots.

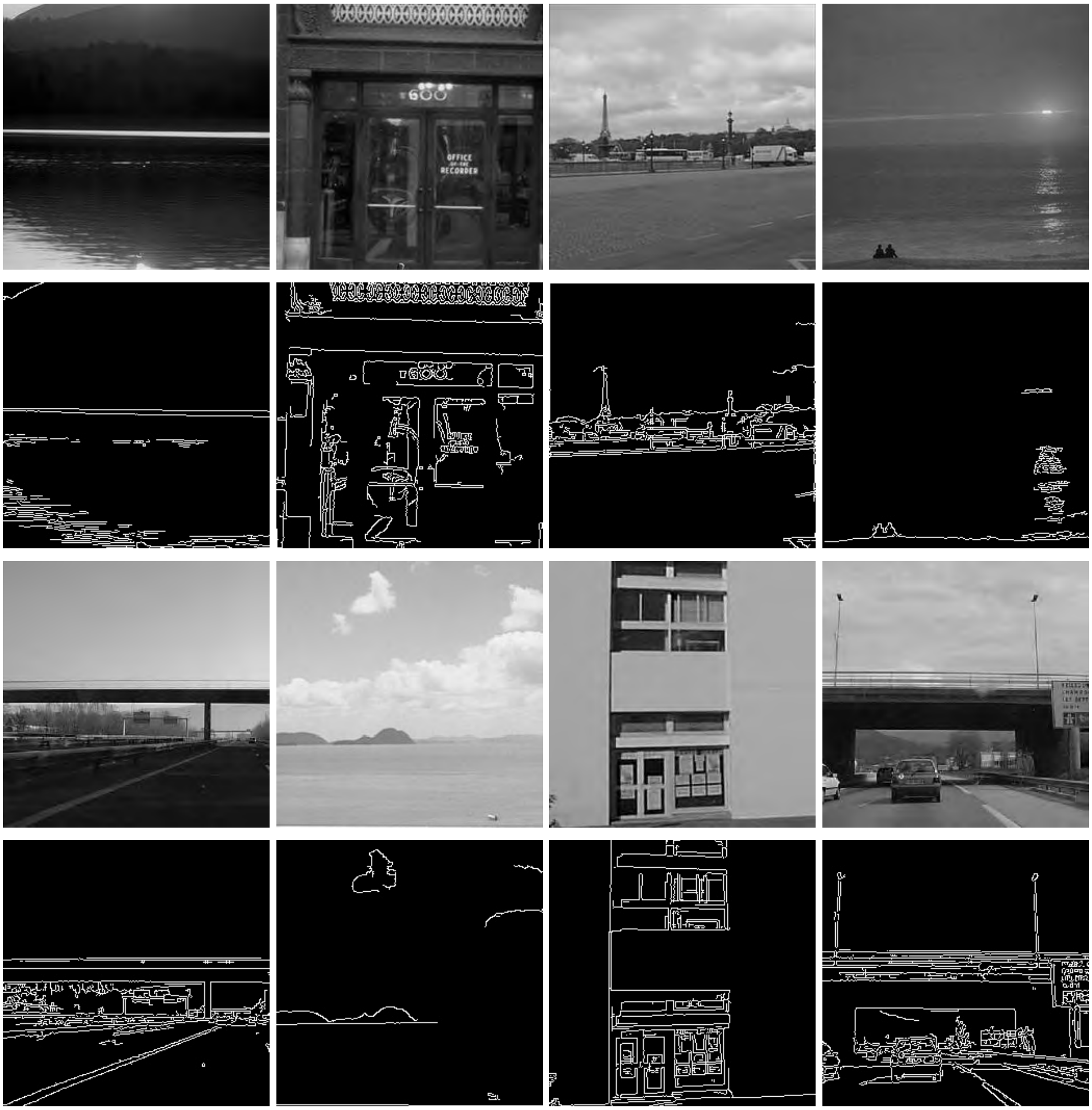


Figure 4.3: False Positives Using LSP Feature Set. This set is of images that were classified as screenshots but were actually non-screenshots. These images tended to fall in two main categories. The first category is images that have very few detected line segments, so even just a few horizontal or vertical line segments would cause a noticeable impact in the percentage (since the denominator in the percentage calculation is so low). The second category is of non-screenshots that actually happen to have a proportionally large number of lines that fall within our threshold for qualifying a line segment as horizontal or vertical.

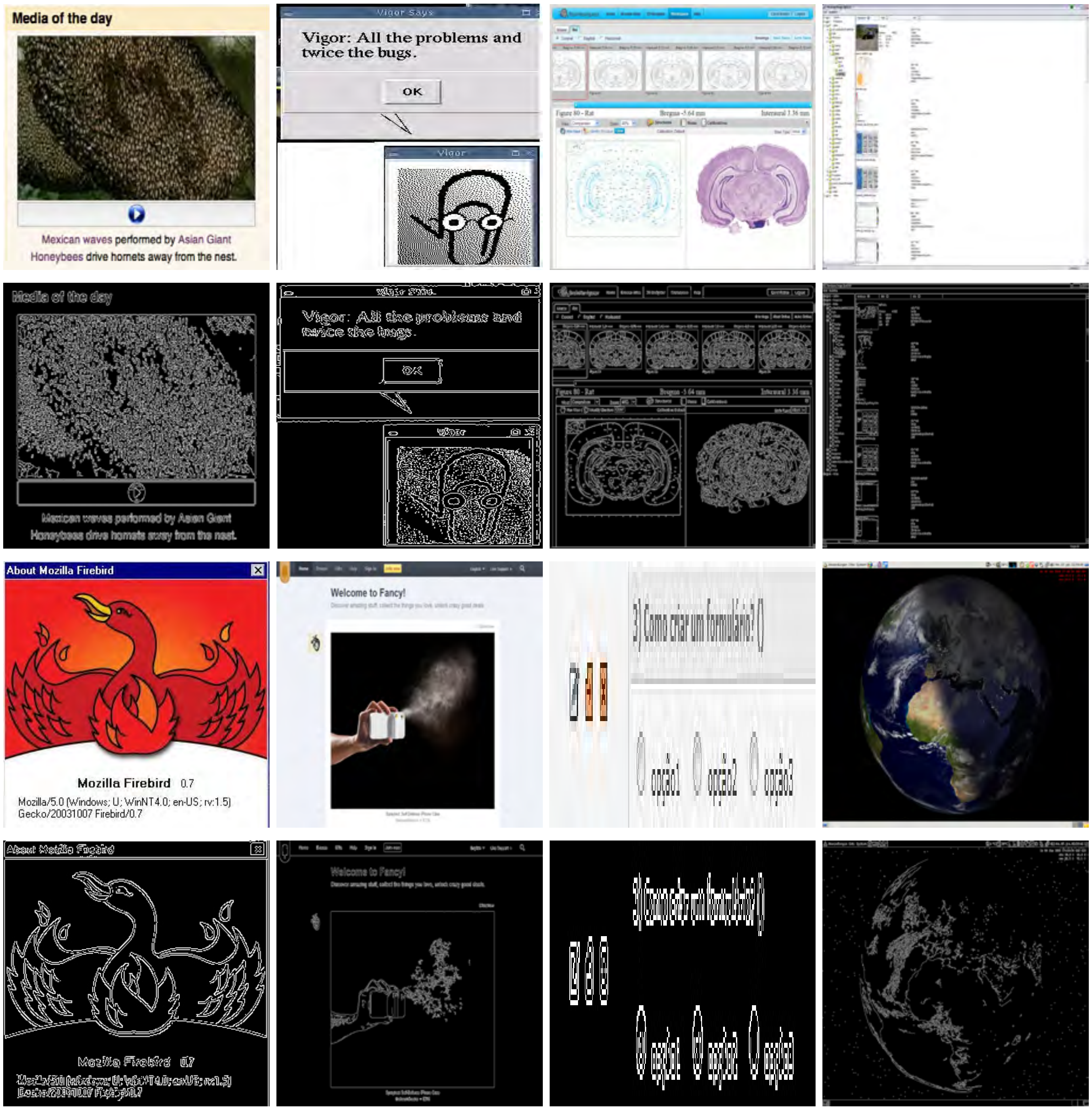
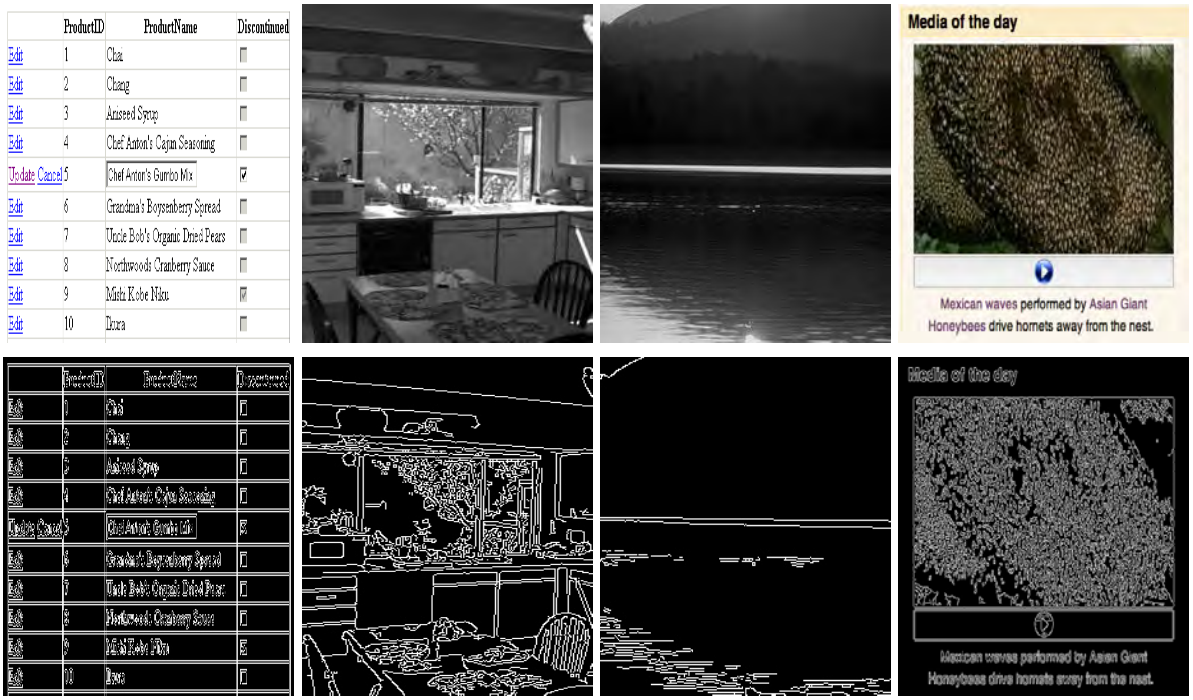

Media of the day
Mexican waves performed by Asian Giant Honeybees drive hornets away from the nest.
Vigor Says
Vigor: All the problems and twice the bugs.
OK
Vigor
About Mozilla Firebird
Mozilla Firebird 0.7
Mozilla/5.0 (Windows; U; WinNT4.0; en-US; rv:1.5) Gecko/20031007 Firebird/0.7
Welcome to Fancy!
3) Como criar um formulário? ()
opção1
opção2
opção3

Figure 4.4: False Negatives Using LSP Feature Set. This set of images is of screenshots that were misclassified (the algorithm labeled them as non-screenshots instead). Even though it is clear to the human observer that they are indeed screenshots, these examples have a number of line segments that would not be counted as horizontal or vertical and the quantity of these line segments is enough to outweigh the horizontal and vertical line segments are do exist in the images.



	ProductID	ProductName	Discontinued
Edit	1	Chai	□
Edit	2	Chang	□
Edit	3	Aniseed Syrup	□
Edit	4	Chef Anton's Cajun Seasoning	□
Update Cancel	5	Chef Anton's Gumbo Mix	☑
Edit	6	Grandma's Boysenberry Spread	□
Edit	7	Uncle Bob's Organic Dried Pears	□
Edit	8	Northwoods Cranberry Sauce	□
Edit	9	Mishi Kobe Niku	☑
Edit	10	Ikura	□

Mexican waves performed by Asian Giant Honeybees drive hornets away from the nest.

(a) True Positive (b) True Negative (c) False Positive (d) False Negative

Figure 4.5: LSP Example Images. This figure contains one image from each of the four confusion matrix categories enumerated (true positive, true negative, false positive, and false negative) and their corresponding line segment extractions while Table 4.3 contains the feature values of each of the four images and the average features values from the 2400 tested screenshots and the 2400 tested non-screenshots.

	Horizontal Percent	Vertical Percent	Horizontal or Vertical Percent
Average Screenshot	0.750	0.109	0.859
Average Photo	0.179	0.067	0.246
True Positive 4.5(a)	0.777	0.165	0.942
True Negative 4.5(b)	0.156	0.044	0.200
False Positive 4.5(c)	0.615	0.000	0.615
False Negative 4.5(d)	0.193	0.013	0.206

Table 4.3: LSP Example Image Feature Values

4.2 Feature Set 2: LSB

Our second experiment provided a greater level of detail to the previous classifier's use of line segment information as features. Instead of treating all line segments equally (independent of length of the line segment), the second feature set consists of counts of horizontal or vertical lines binned by length (and proportional to the size of the image).

In this experiment, we tested a number of different bin sizes. Instead of providing confusion matrices for each of the different bin sizes, we include the following summary statistics as calculated from the confusion matrices.

Number of Bins	Accuracy	Precision	Recall	F-score
10	0.882	0.977	0.782	0.869
50	0.869	0.989	0.746	0.850
100	0.873	0.993	0.751	0.855
500	0.887	1.000	0.774	0.872
1000	0.891	0.999	0.783	0.878

Table 4.4: LSB Results as a Function of Line Length Bins

Within the scope of the LSB feature set, varying the number of line segment length bins tended to have a varying effect on the results (as shown in Table 4.4). Most consistent is the fairly-steady improvement of the algorithm's precision (due to the reduction of false positives) as the number of bins increases. When comparing the LSB experiment to the LSP experiment, the LSB experiment provides a higher precision and lower accuracy independent of the number of bins used. The following two sets of images demonstrate the improvements that binning provides. For these examples, we decided to use the results from the LSB feature set with 1000 bins because it outperformed (albeit negligibly) the other LSB bin sizes with respect to f-score. Figure 4.6 contains eight non-screenshot images that were incorrectly classified as screenshots when solely using the LSP feature set, but are correctly classified using only the LSB feature set. Figure 4.7 presents a set of eight screenshots that were misclassified by LSP, but correctly classified by LSB.

4.2.1 Example Images

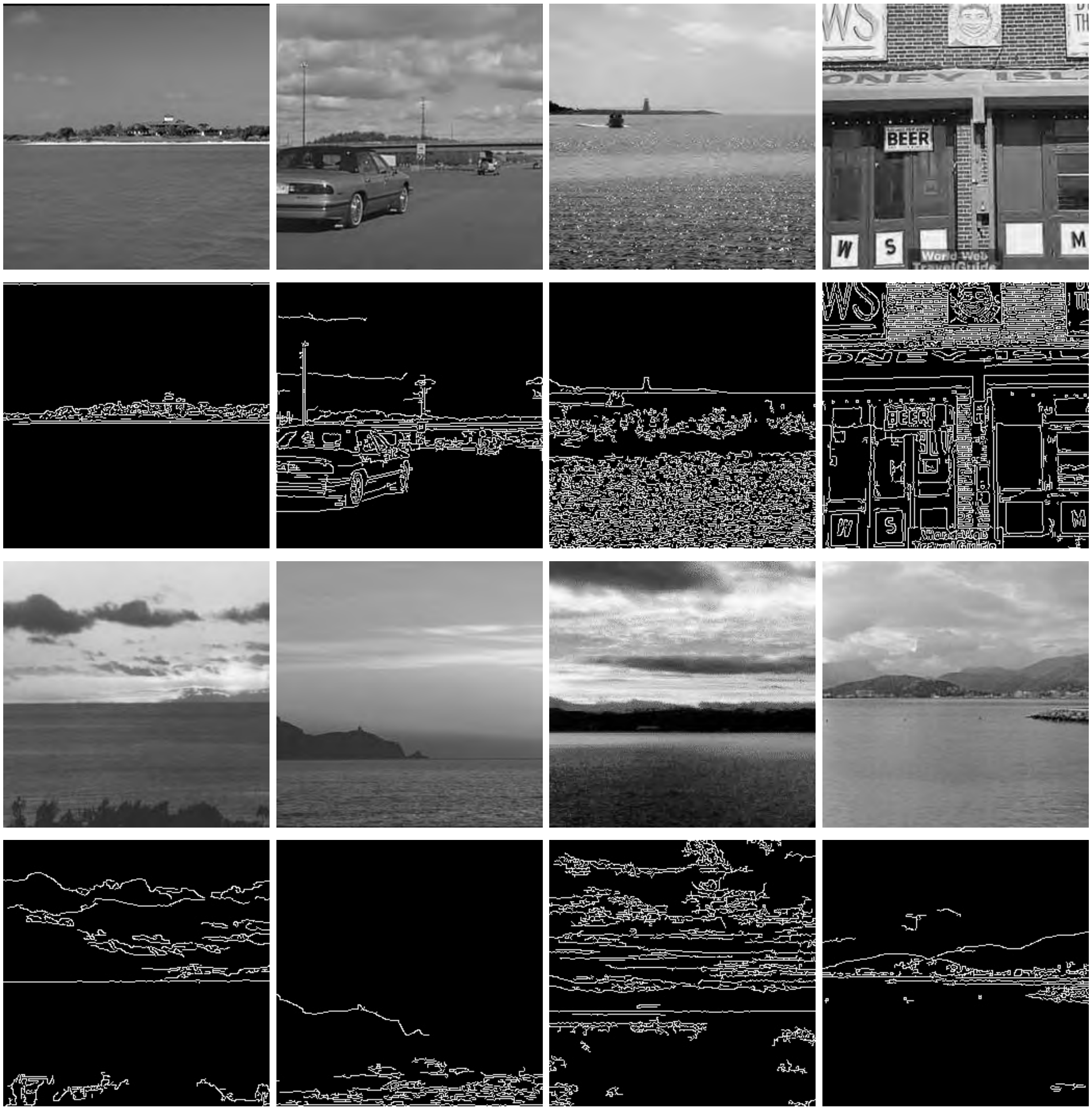


Figure 4.6: True Negatives Using LSB Feature Set with 1000 Bins. The LSP features were frequently unsuccessful in correctly classifying non-screenshots that contained a small number of line segments if a few of those segments were horizontal or vertical (as is often the case in images of horizons). Because all three features were percentage-based, the small number of total line segments allowed a few horizontal or vertical segments to significantly influence the feature values. The binned features improved upon this by employing features based on counts as opposed to percentages.

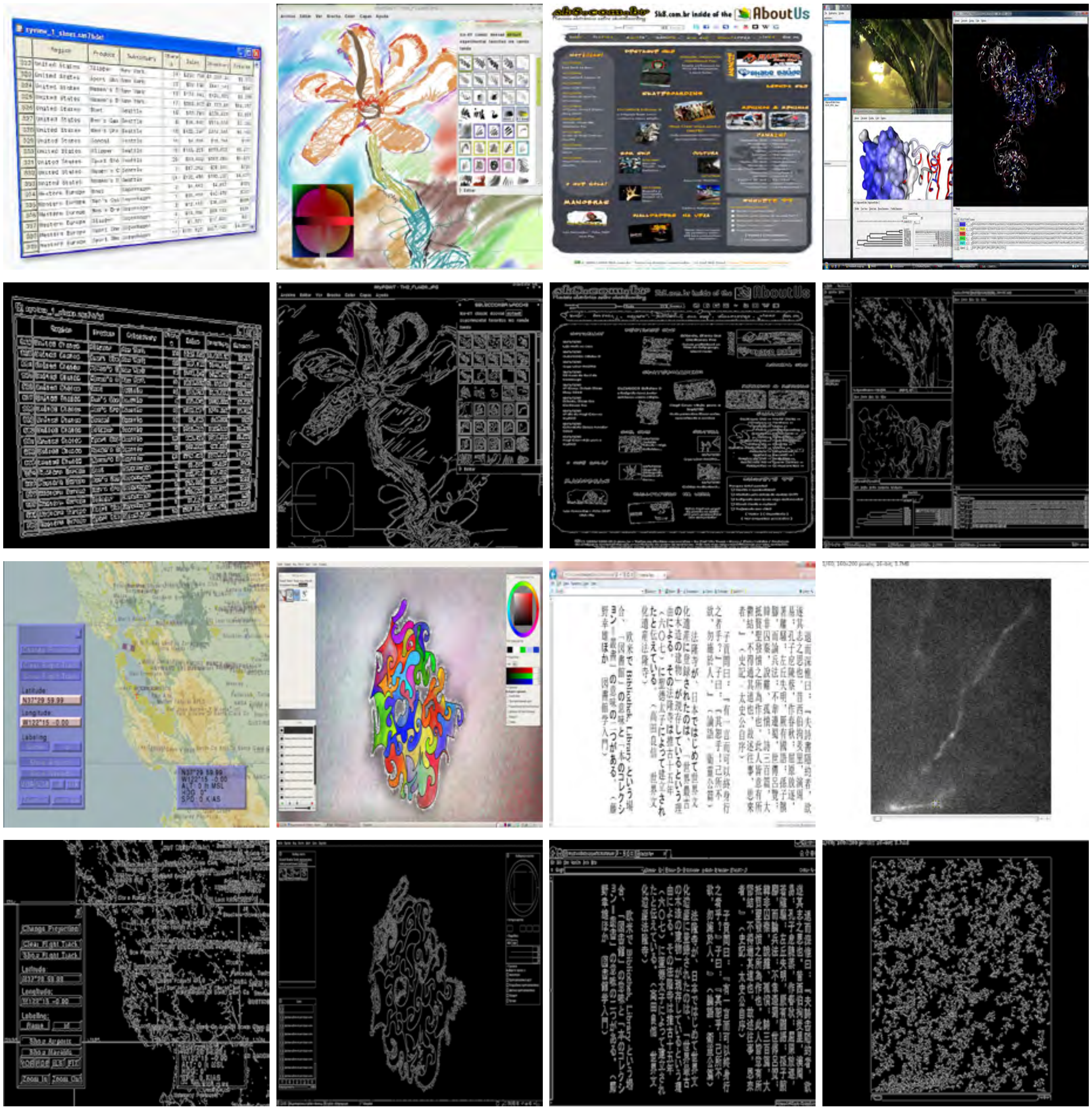


Figure 4.7: True Positives Using LSB Feature Set with 1000 Bins. Although the number of true positives decreased between the LSB feature set and the LSP feature set, there were some screenshots for which the LSB feature set was better suited. Line segment binning seemed to be effective in correctly classifying screenshots that have significant clutter (in their collection of line segments) but that also contain noticeable linear features of relatively large length, usually from screen elements like menu bars.

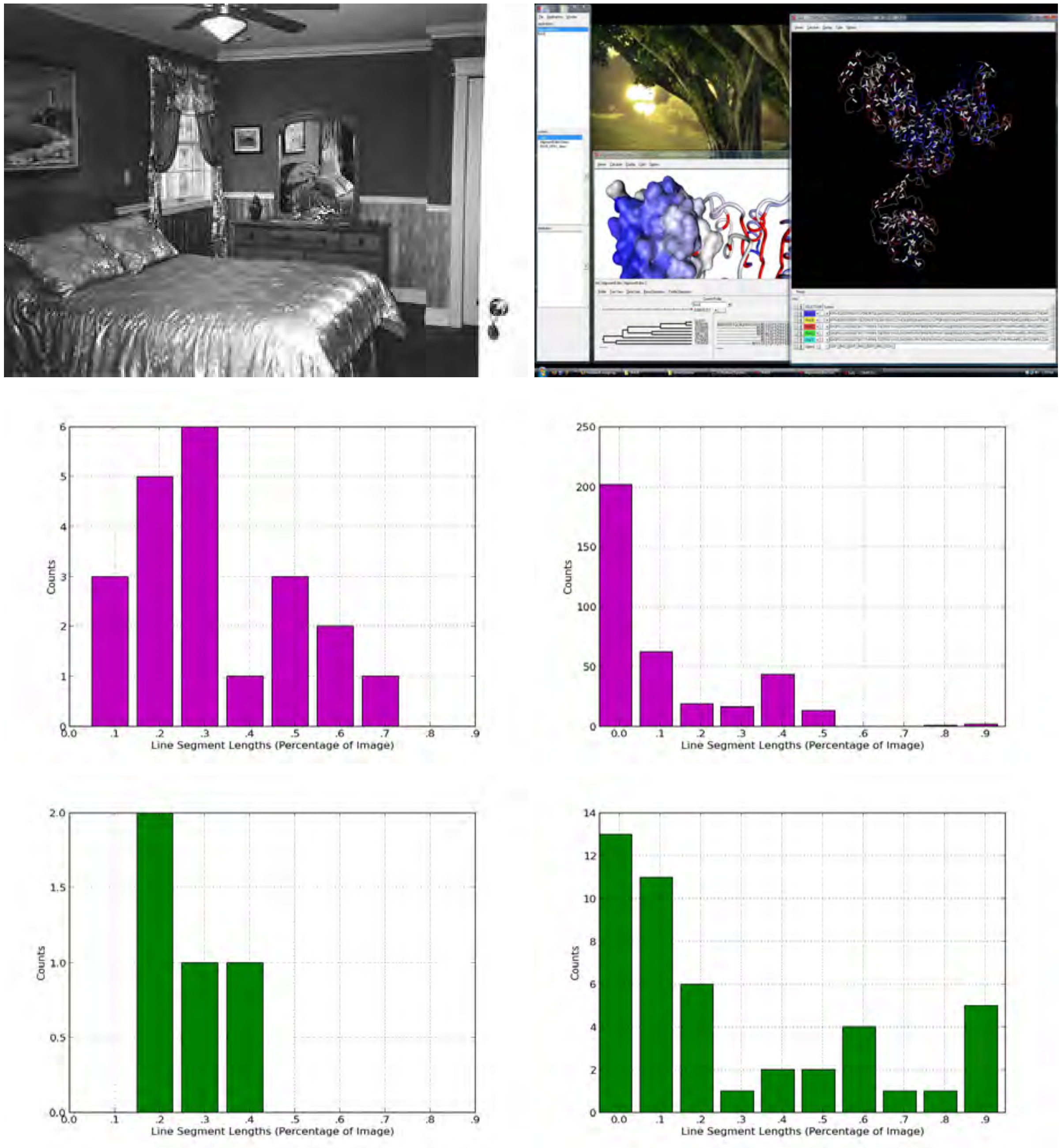


Figure 4.8: LSB Feature Values (Ten Bins) for Correctly Classified Images. The second row is the horizontal line segments, while the third row is the vertical line segments.

4.3 Feature Set 3: CE

The third experiment used intensity-based featured instead of linear-based features. Similar to the LSP and LSB feature sets, we started with a simpler set of features, or in this case one feature—intensity entropy.

	Predicted Screenshot	Predicted Non-screenshot
Actual Screenshot	2044	356
Actual Non-screenshot	98	2302

Table 4.5: Experiment 3 Confusion Matrix

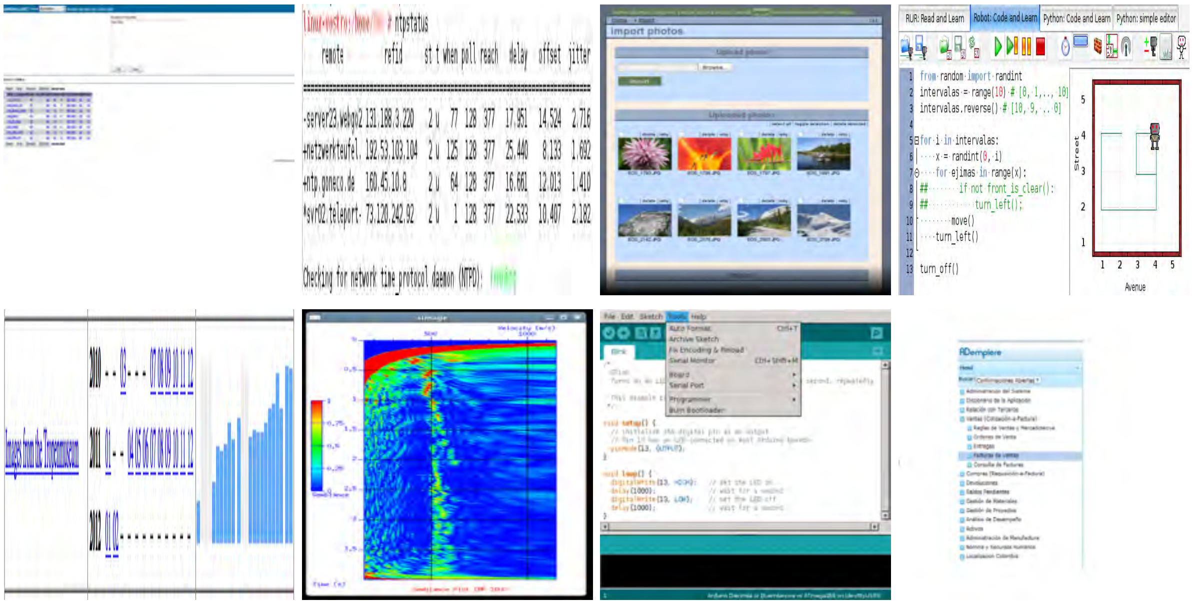
Table 4.5 presents the confusion matrix for the experiment utilizing only the feature of entropy. Additionally, summary statistics (as calculated from the confusion matrix) are listed in Table 4.6.

Accuracy	Precision	Recall	F-score
0.905	0.954	0.852	0.900

Table 4.6: CE Results

4.3.1 Example Images

Figures 4.9–4.12 contain a random sampling of true positives, true negatives, false positives, and false negatives from this experiment. Due to the use of entropy as a singular feature in this experiment, the results are as expected. Images with a more even distribution of different grayscale pixel values have a higher entropy and are more likely to be a non-screenshot. Computer-generated images, on the other hand, tend to have lower entropy values due to uneven distributions of pixel values, and large concentrations of the same pixel value within an image.



```
linux-vostro:/home/ # ntpstatus
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
-server23.webgo2 131.188.3.220    2 u   77  128  377   17.951   14.524   2.716
+netzwerkteufel. 192.53.103.104   2 u  125  128  377   25.440    8.133   1.692
+ntp.goneco.de   160.45.10.8      2 u   64  128  377   16.661   12.013   1.410
*svr02.teleport- 73.120.242.92    2 u    1  128  377   22.533   10.407   2.182

Checking for network time protocol daemon (NTPD):
```

```
from random import randint
intervalas = range(10) # [0, 1,.., 10]
intervalas.reverse() # [10, 9, .. 0]

for i in intervalas:
    x = randint(0, i)
    for ejimas in range(x):
##        if not front_is_clear():
##            turn_left();
        move()
    turn_left()

turn_off()
```

Figure 4.9: True positives Using CE Feature Set

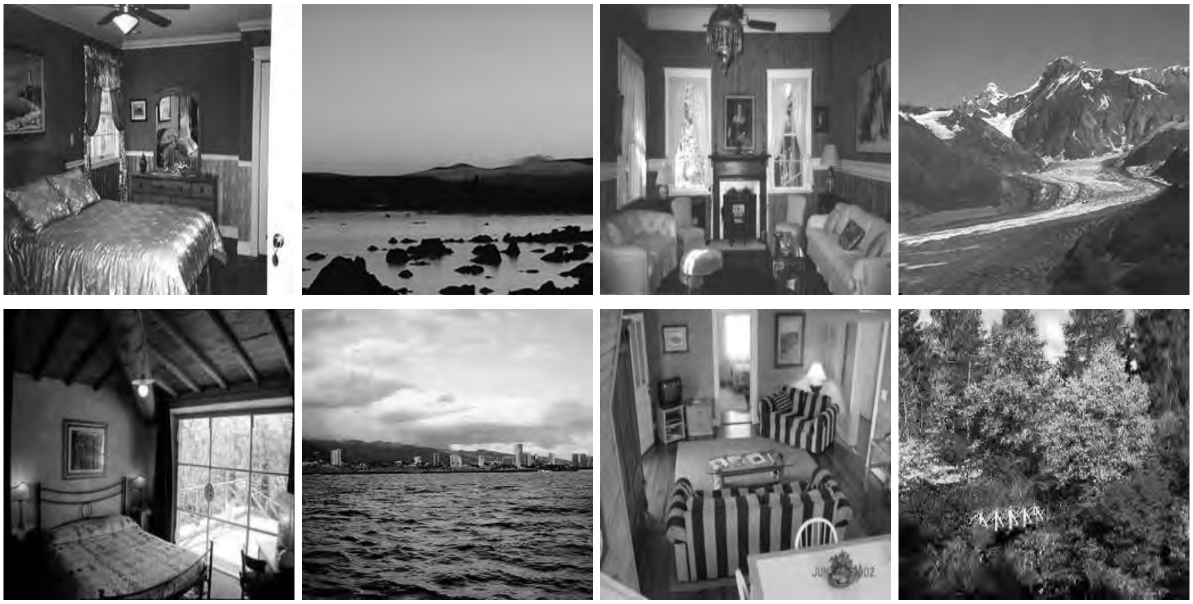


Figure 4.10: True Negatives Using CE Feature Set

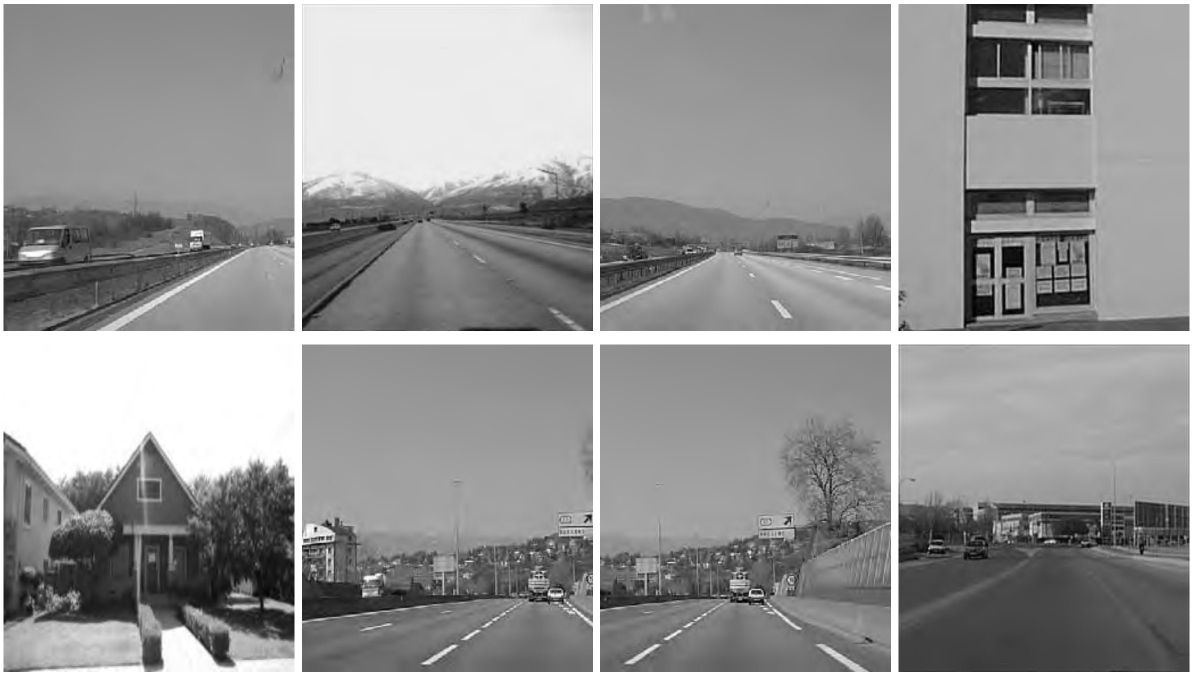


Figure 4.11: False Positives Using CE Feature Set

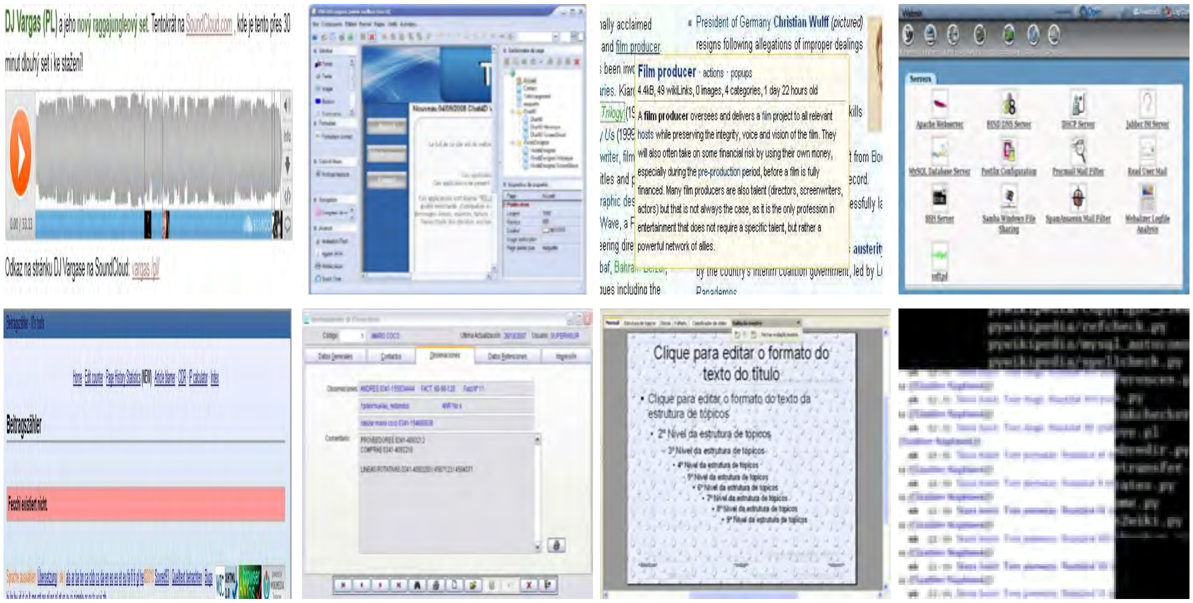
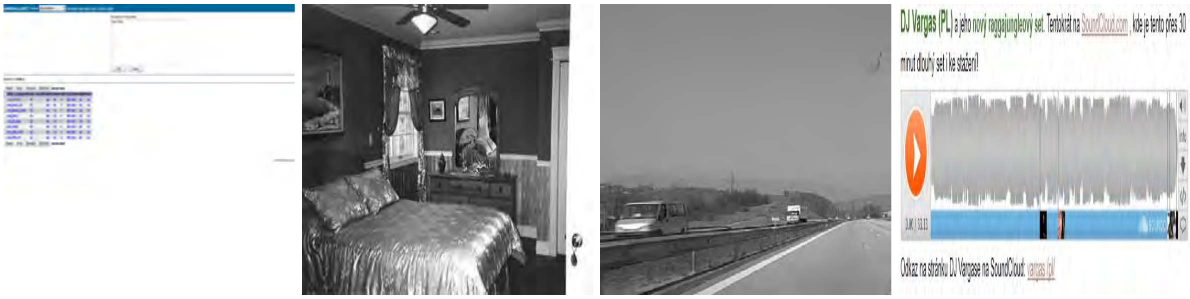


Figure 4.12: False Negatives Using CE Feature Set

A sample image from each category is shown in Figure 4.13 and their respective entropy values in Table 4.7. While the entropies of the correctly classified images (true positive and true negative) fall close to their categorical averages or even on the outside of the average (away from the other class mean), the entropies of the misclassified images are closer to each other and both between the two class means.



DJ Vargas (PL) a jeho nový raggajungleový set. Tentokrát na SoundCloud.com , kde je tento přes 30 minut dlouhý set i ke stažení!

Odkaz na stránku DJ Vargase na SoundCloud: vargas /pl/

(a) True Positive: 2.356 (b) True Negative: 7.283 (c) False Positive: 6.077 (d) False Negative: 6.634

Figure 4.13: CE Example Images and Entropy Values

	Entropy
Average Screenshot	4.878
Average Non-screenshot	7.336
True Positive 4.13(a)	2.356
True Negative 4.13(b)	7.283
False Positive 4.13(c)	6.077
False Negative 4.13(d)	6.634

Table 4.7: LSP Example Image Feature Values

4.4 Feature Set 4: CB

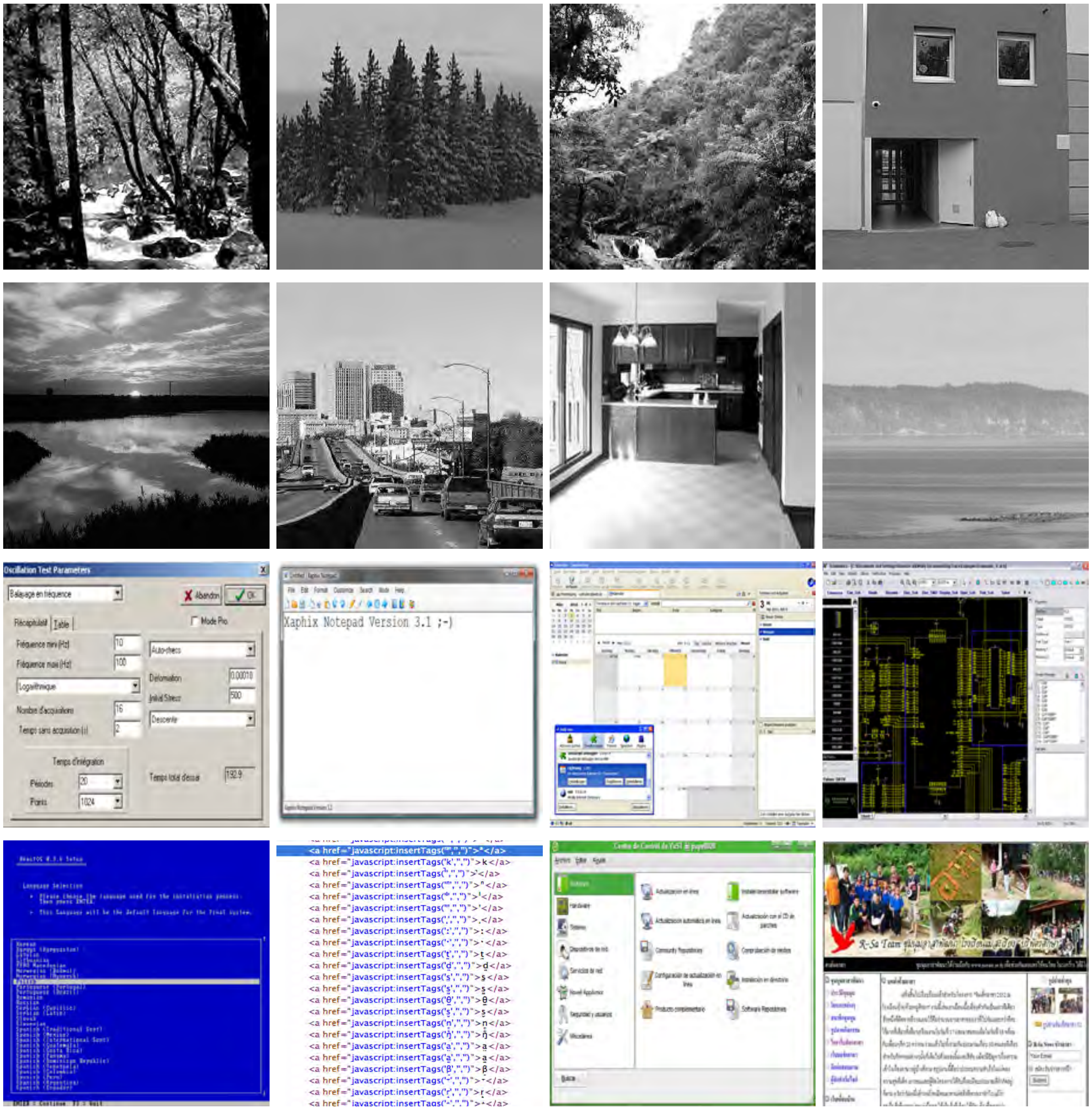
Similar to the evolution of linear-based features, we explored whether a more detailed representation of an image's pixel-intensity distribution would increase the success of the classification algorithm (as compared to the CE feature set). The details of the implementation of this feature set are described in detail in the previous chapter. The CB feature set also requires a parameter for the number of bins in which to place the color histogram bin heights. We tested a variety of bin sizes between ten and 5,000. Unlike the LSB feature set, bin size played a significant role in the success of the algorithm (as shown in Table 4.8). As bin size increased, the values for accuracy, precision, recall, and F-score almost always increased.

Number of Bins	Accuracy	Precision	Recall	F-score
10	0.773	0.947	0.578	0.717
50	0.802	0.847	0.736	0.788
100	0.816	0.863	0.751	0.803
500	0.945	0.979	0.910	0.943
1000	0.945	0.981	0.910	0.944

Table 4.8: Results as a Function of Color Bins

4.4.1 Example Images

Figure 4.14 demonstrates this feature set's progressive increase in success as the number of bins increases. The first column of the figure contains two screenshots and two non-screenshots that were incorrectly classified when the number of bins was ten, but became correctly classified when the number of bins was increased to 50. The second column shows four images that made this transition when the number of bins was increased again from 50 to 100; the rest of the columns follow the same pattern.


Oscillation Test Parameters
Xaphix Notepad Version 3.1 ;-)
Centro de Control de YaST

(a) Bins=50 (b) Bins=100 (c) Bins=500 (d) Bins=1000

Figure 4.14: Correctly Classified Images as Bin Size Increase. Following the progress from left to right in this figure, the images increase in classification "difficulty." That is, the colors in the screenshots become more varied and the colors in the non-screenshots become more monotonous. As bin size increased, the performance of the algorithm also increased.

4.5 Feature Set Combinations

After running the experiments on each of the individual feature sets, we ran the experiment on all possible combinations of the four feature sets. In cases where the feature set required a parameter for the number of bins (LSB and CB), five bin sizes were tested (10, 50, 100, 500 and 1000). The accuracy, precision, recall and F-score of the eleven additional experiments are listed in Table 4.9, for the full results, please see the appendix. For experiments with bin number parameters, only the best bin size combination is presented in this chart (as measured by highest F-score).

Set 1	Set 2	Set 3	Set 4	Line Bins	Color Bins	Accuracy	Precision	Recall	F-score
–	–	Yes	Yes	–	50	0.942	0.950	0.933	0.941
–	Yes	–	Yes	10	500	0.968	0.993	0.942	0.967
–	Yes	Yes	–	10	–	0.937	0.994	0.880	0.933
–	Yes	Yes	Yes	10	500	0.966	0.991	0.941	0.965
Yes	–	–	Yes	–	500	0.963	0.989	0.937	0.962
Yes	–	Yes	–	–	–	0.954	0.963	0.945	0.954
Yes	–	Yes	Yes	–	10	0.967	0.981	0.953	0.967
Yes	Yes	–	–	10	–	0.956	0.973	0.937	0.955
Yes	Yes	–	Yes	50	500	0.975	0.997	0.952	0.974
Yes	Yes	Yes	–	10	–	0.973	0.987	0.957	0.972
Yes	Yes	Yes	Yes	10	10	0.980	0.997	0.963	0.980

Table 4.9: Best Results by Feature Set Combination

In Table 4.9 we can see that the majority of experiments using the LSB feature set showed best results when the number of bins was set to the lowest value, ten. Also, for the most part, the inclusion of additional feature sets noticeably increased the success of the algorithm.

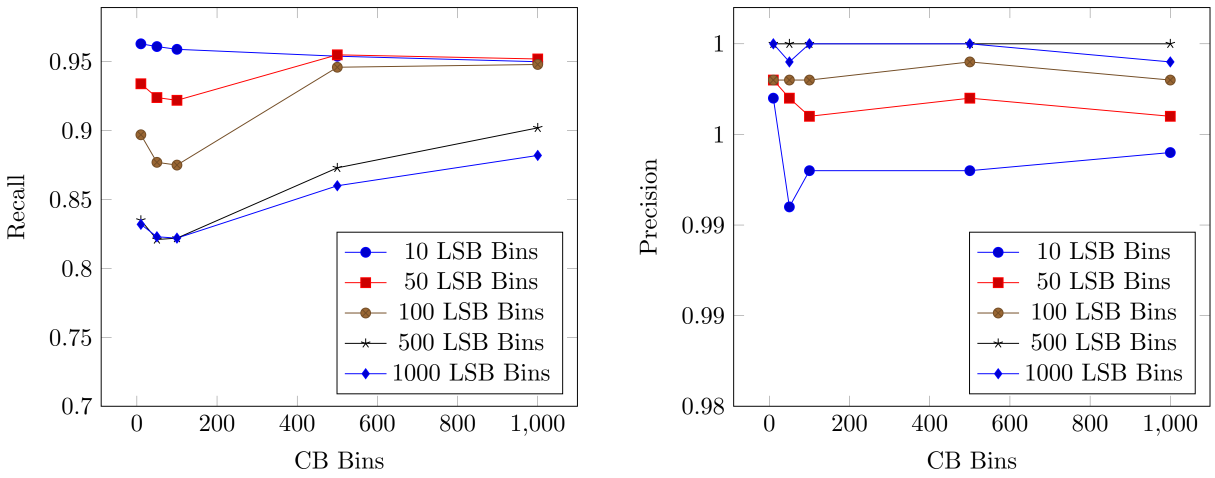


Figure 4.15: Results Using All Four Feature Sets. Each line represents a specific value of the LSB bins parameter (as noted in the key) and moving across the line from left to right indicates and increase the CB bins parameter.

Figure 4.15 presents two graphs for the experiment that contains all four feature sets. These graphs present accuracy and precision as a function of the number of line bins and color bins from the LSB and CB feature sets. We only present graphs for recall and precision, as opposed to all four major summary statistics, because the graphs for accuracy and F-score mirror the trends in the recall graph, only with slightly scaled values. As a note, the y-axis scales for the graphs are different because the values for precision tended to be significantly higher than those of the other metrics. Precision values all fell between 0.990 and 1.000 regardless of the values chosen for LSB bins and CB bins. The trade-off between recall and precision can be manipulated using the prior probability of a class within the Bayesian classifier calculations. For our experiments, the prior probability was determined by the number of images of each class in our training set, which was evenly split between screenshots and non-screenshots. The recall-precision trade-off is then a function of the classifier's ability to more easily discern one class than the other. Figure 4.16 shows the receiver operating characteristic (ROC) curve for the classifier learned from the LSP feature set. The diagonal line in the plot shows the curve of a classifier that guesses at random while an ideal curve puts all the points in the upper left corner. The ROC curve for the classifier learned using all four feature sets is virtually identical to the ideal curve (hence why it isn't shown).

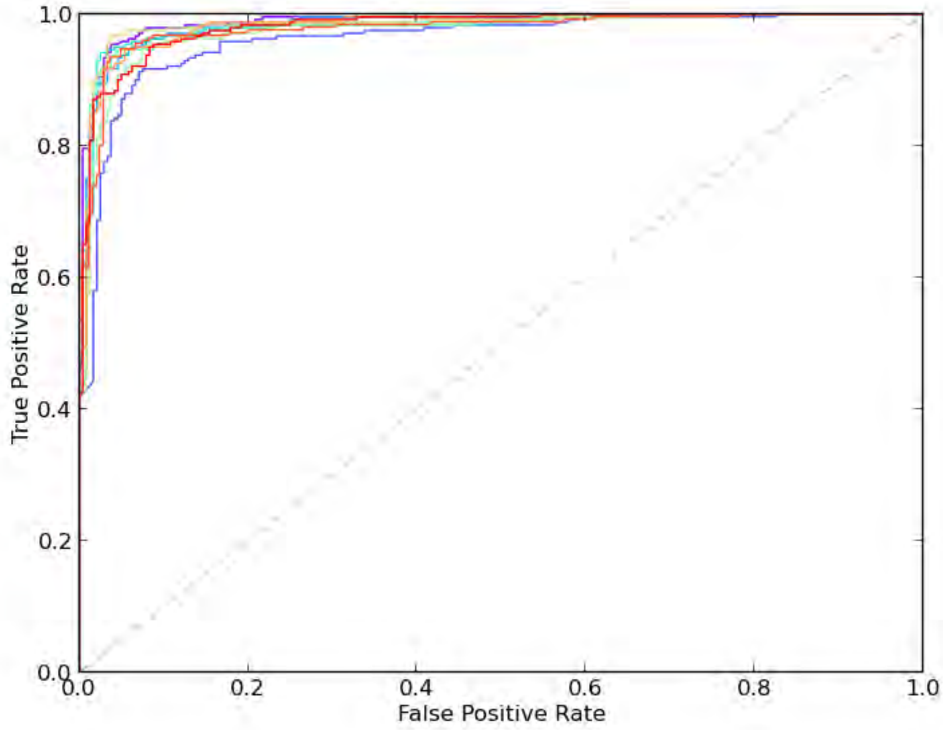


Figure 4.16: ROC Curve for LSP Feature Set. As an example of the trade-offs between false positives and false negatives, we present a ROC curve for one of our experiments (using the LSP feature set). Each line on this graph represents the ROC curve results for a single iteration of the ten-fold cross-validation.

4.6 Computational Performance

The purpose of this thesis was not to measure the speed of the algorithm, but we provide rough estimations of the time performance of the algorithm. The algorithm was tested on an Ubuntu 32-bit Virtual Machine with 1GB RAM which was running within VMware Player on Windows 7 with 8GB RAM and an Intel Core i7-2600 processor. The algorithm that combined all four feature sets (with bin size parameter values of ten for both the LSB and CB feature sets) was able to calculate the features for 4800 images, and train and test the classifier in about 14 minutes. This averages to about 0.18 seconds per image including training and testing.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Conclusions and Future Work

In the previous chapters, we presented our motivation, experimental design and results regarding an evaluation of feature sets for a computer screenshot detection algorithm. As with most research projects and especially those that address tangible and evolving real-world problems, there is always room for improvement, expansion, and future work. There are a number of areas ripe with additional work, and we will address three that particularly stand out.

5.1 Feature Set Improvements

The first area for future work is the continued refinement of the feature sets for screenshot detection. We selected four feature sets, varied their parameters, and tested combinations of feature sets. While there is more work to be done in fine-tuning the four feature sets from this paper, there are a number of potential features that we did not address. We focused on low-level (pixel-based) features. Two additional low-level features are one- and two-dimensional run-length encodings of pixel values in order to quantify the groupings of identically-colored pixels. At a more abstract level, features such as logo detection (for example, being able to identify the Windows start button) and optical character recognition (determining if the image has text content) could provide indications as to the class of image.

At an even higher level, there is often distinguishing metadata associated with digital images. Mac's OS X operating system automatically gives standard filenames to screenshots taken through the standard screen capture keystroke sequence, whereas some photographs taken with digital cameras store metadata about everything from the camera make and model to the ISO speed and focal length. The aspect ratio of an image can also give an indication of the origin of that image. While this information can certainly be forged or tampered with, it could still provide improved classification success.

5.2 Efficiency-Effectiveness Trade-Off

When developing a machine learning classification algorithm, more features often provide better classification success, but it can be difficult to accurately measure the time and computational costs associated with each additional feature or feature set. It would be useful future work to complete a study comparing the inverse relationship between efficiency and effectiveness as it pertains to the number of features used in a screenshot classification algorithm and the speed with which the algorithm can perform. This type of study would be particularly relevant for at least two reasons: tools that assist in forensic triage require optimized speed and performance, and that the specific problem of distinguishing screenshots from non-screenshots may be better suited for a streamlined feature set than the task of a more general image categorization algorithm.

5.3 Requirements Analysis

Finally, a key area for future work is the process of understanding what an actual user would require for this system to be useful. Information about its intended use would also assist in parameter tuning and help to optimize the algorithm for its target audience. As with any classification algorithm, there is a trade-off between precision and recall that can be adjusted by changing any number of parameters. By performing requirements analysis with a potential future user of this algorithm (for example a forensic examiner), it would be possible to develop a use case and determine what number of errors the user can tolerate and more importantly, whether they prefer false positives or false negatives. A tool implementing this algorithm could also provide the end user the ability to perform real-time tuning to alter this trade-off.

5.4 Performance Example

If a forensic examiner was presented with a 1 terabyte drive completely filled with images, that drive would be able to hold about 525,000 images if each image was 2 megabytes in size. Assuming we achieve the same accuracy results as our best-performing experiment (.98), that means that just over 10,000 of the images on the drive would be wrongly classified. It is here that the precision-recall trade-off described earlier has an effect. If an examiner needs to find as many screenshots as possible and is able to tolerate sifting through more non-screenshots to do that, then the algorithm can be tuned to retrieve more images total, thereby increasing the number of false positives and decreasing precision. On the other hand, if an examiner has very little time to examine the drive and does not have the workload capacity to sift through unnecessary non-screenshots in order to find screenshots, fewer images can be returned to the examiner as potential screenshots. This alternative would increase the number of false negatives (screenshots that were incorrectly classified as non-screenshots) as well as decreasing recall.

5.5 Conclusion

In our research, we successfully designed, built, and tested features for screenshot detection. We developed four feature sets that captured the linear and intensity-based characteristics, which we originially hypothesized would be applicable in distinguishing a screenshot from a non-screenshot. We tested various combinations of all feature sets and parameter values and were able to measure aspects of their success through the metrics of accuracy, precision, recall, and F-score obtained from the ten-fold cross-validation. Some of our more minimal features sets had F-scores between 0.96 and 0.98 and our most promising feature set combination returned results with an F-score of 0.98. After determining the best combination of our feature sets, we are able to recommended a successful screenshot detection algorithm that provides a valuable, and otherwise unavailable triage tool to today's forensic examiners.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] rlittle. (2012, Apr.) Ff 12.0 saves images of each webpage i visit to my cache folders. is this a feature or malware? [Online]. Available: https://support.mozilla.org/en-US/questions/926354

[2] B. Reid. (2012, Mar.) New mac os x malware discovered, takes screenshots and uploads them to unknown servers without user's consent. [Online]. Available: http://www.redmondpie.com/new-mac-os-x-malware-discovered-takes-screenshots-and-uploads-them-to-unknown-servers-without-user-consent/

[3] H. Moravec, *Mind Children: The Future of Robot and Human Intelligence*. Cambridge, MA: Harvard University Press, 1988 1988.

[4] Ibm100 - deep blue. [Online]. Available: http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/

[5] R. Szeliski, *Computer Vision: Algorithms and Applications (Texts in Computer Science)*, 1st ed. Springer, Nov. 2010. [Online]. Available: http://www.worldcat.org/isbn/1848829345

[6] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *CVPR*, 2010, pp. 3485–3492.

[7] L. Fei-Fei and P. Perona, "A Bayesian hierarchical model for learning natural scene categories," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, June 2005, pp. 524 – 531 vol. 2.

[8] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94

[9] C. Schmid, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *In CVPR*, 2006, pp. 2169–2178.

[10] S. N. Parizi, J. G. Oberlin, and P. F. Felzenszwalb, "Reconfigurable models for scene recognition," in *CVPR*, 2012, pp. 2775–2782.

[11] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, pp. 145–175, 2001.

[12] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision Image Understanding*, vol. 110, no. 3, pp. 346–359, Jun. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2007.09.014

[13] M. J. Swain and D. H. Ballard, “Color indexing,” *International Journal of Computer Vision*, vol. 7, pp. 11–32, 1991.

[14] A. Quattoni and A. Torralba, “Recognizing indoor scenes,” in *CVPR*, 2009, pp. 413–420.

[15] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital Image Processing Using MATLAB, 2nd ed.*, 2nd ed. Gatesmark Publishing. [Online]. Available: http://www.worldcat.org/isbn/0982085400

[16] L. G. Roberts, “Machine perception of three-dimensional solids,” DTIC Document, Tech. Rep., 1963.

[17] J. Canny, “A computational approach to edge detection,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-8, no. 6, pp. 679 –698, nov. 1986.

[18] J. Prewitt, *Object enhancement and extraction.* Academic Press, New York, 1970, vol. 75.

[19] J. B. Burns, A. R. Hanson, and E. M. Riseman, “Extracting straight lines,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 4, pp. 425–455, 1986.

[20] P. Hough, “Method and means for recognizing complex patterns,” December 1962, US Patent 3,069,654.

[21] R. M. Haralick, “Ridges and valleys on digital images,” *Computer Vision, Graphics, and Image Processing*, vol. 22, no. 1, pp. 28–38, 1983.

[22] M. A. Fischler, “Linear delineation,” DTIC Document, Tech. Rep., 1982.

[23] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, “LSD: a Line Segment Detector,” *Image Processing On Line*, 2012.

[24] A. Halder, N. Chatterjee, A. Kar, S. Pal, and S. Pramanik, “Edge detection: A statistical approach,” in *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, vol. 2, April 2011, pp. 306 –309.

[25] OpenCV. (2013, Apr.) Opencv). [Online]. Available: http://opencv.org/

[26] S. Hinds, J. Fisher, and D. D’Amato, “A document skew detection method using run-length encoding and the hough transform,” in *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, vol. i, June 1990, pp. 464 –468 vol.1.

[27] T. M. Mitchell, *Machine Learning*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997.

[28] D. Michie, D. J. Spiegelhalter, C. C. Taylor, and J. Campbell, “Machine learning, neural and statistical classification,” 1994.

[29] S. Aksoy, K. Koperski, C. Tusk, G. Marchisio, and J. Tilton, “Learning bayesian classifiers for scene classification with a visual grammar,” *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 43, no. 3, pp. 581 – 589, March 2005.

[30] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995. [Online]. Available: http://dx.doi.org/10.1023/A:1022627411411

[31] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines: and other kernel-based learning methods*. New York, NY, USA: Cambridge University Press, 2000.

[32] O. Chapelle, P. Haffner, and V. Vapnik, “Support vector machines for histogram-based image classification,” *Neural Networks, IEEE Transactions on*, vol. 10, no. 5, pp. 1055 –1064, September 1999.

[33] J. Hao and X. Jie, “Improved bags-of-words algorithm for scene recognition,” in *Signal Processing Systems (ICSPS), 2010 2nd International Conference on*, vol. 2, July 2010, pp. V2–279 –V2–282.

[34] (2013, Apr.) Category:screenshots - wikimedia commons. [Online]. Available: http://commons.wikimedia.org/wiki/Category:Screenshots

[35] pythonware. (2013, Apr.) Python imaging library (pil). [Online]. Available: http://www.pythonware.com/products/pil/

[36] NumPy. (2013, Apr.) Scientific computer tools for python - numpy). [Online]. Available: http://www.numpy.org/

[37] T. Curk, J. Demšar, Q. Xu, G. Leban, U. Petrovič, I. Bratko, G. Shaulsky, and B. Zupan, “Microarray data mining with visual programming,” *Bioinformatics*, vol. 21, pp. 396–398, Feb. 2005. [Online]. Available: http://bioinformatics.oxfordjournals.org/content/21/3/396.full.pdf

[38] W. S. Cleveland and S. J. Devlin, “Locally-Weighted Regression: An Approach to Regression Analysis by Local Fitting,” *Journal of the American Statistical Association*, vol. 83, pp. 596–610, 1988.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix: Detailed Results

In order to avoid verbosity when describing each of the experiments, we created a succinct four-part naming schema that represents the distinguishing parameters of each experiment.

- `[<number of images>images]` from each class of image
- `[<four-digit binary string>]` representing feature set inclusion
  - `[0/1]` feature set one: LSP
  - `[0/1]` feature set two: LB
  - `[0/1]` feature set two: CE
  - `[0/1]` feature set two: CB
- `[<number of line bins>lbins]` if feature set two is included
- `[<number of color bins>cbins]` if feature set four is included

These four items are concatenated together separated by underscores. As an example, the experiment that is tested on 2400 images of each class and includes feature sets one, three, and four (and has 500 color bins for features set four) would result in the experiment string `2400images_1011_500cbins`.

Results From All Experiments

Experiment	**Accuracy**	**Precision**	**Recall**	**F-score**
2400images_0001_1000cbins	0.944	0.978	0.908	0.942
2400images_0001_100cbins	0.814	0.862	0.749	0.801
2400images_0001_10cbins	0.772	0.946	0.577	0.717
2400images_0001_500cbins	0.944	0.978	0.909	0.942
2400images_0001_50cbins	0.794	0.846	0.72	0.778
2400images_0010	0.905	0.954	0.851	0.900
2400images_0011_1000cbins	0.938	0.978	0.896	0.935
2400images_0011_100cbins	0.930	0.963	0.894	0.927
2400images_0011_10cbins	0.942	0.977	0.906	0.940
2400images_0011_500cbins	0.936	0.976	0.893	0.933
2400images_0011_50cbins	0.942	0.950	0.933	0.941
2400images_0100_1000lbins	0.891	0.999	0.783	0.878
2400images_0100_100lbins	0.873	0.993	0.751	0.855
2400images_0100_10lbins	0.882	0.977	0.782	0.869
2400images_0100_500lbins	0.887	1.0	0.774	0.872
2400images_0100_50lbins	0.869	0.989	0.746	0.850
2400images_0101_1000lbins_1000cbins	0.923	1.0	0.846	0.916
2400images_0101_1000lbins_100cbins	0.894	0.999	0.788	0.881
2400images_0101_1000lbins_10cbins	0.898	0.999	0.797	0.887
2400images_0101_1000lbins_500cbins	0.915	1.0	0.830	0.907

Continued on next page

Table 1 – *Continued from previous page*

Experiment	**Accuracy**	**Precision**	**Recall**	**F-score**
2400images_0101_1000lbins_50cbins	0.893	0.999	0.786	0.880
2400images_0101_100lbins_1000cbins	0.962	0.998	0.926	0.960
2400images_0101_100lbins_100cbins	0.880	0.995	0.765	0.865
2400images_0101_100lbins_10cbins	0.895	0.997	0.792	0.883
2400images_0101_100lbins_500cbins	0.951	0.999	0.903	0.949
2400images_0101_100lbins_50cbins	0.880	0.996	0.763	0.864
2400images_0101_10lbins_1000cbins	0.967	0.993	0.941	0.966
2400images_0101_10lbins_100cbins	0.903	0.982	0.821	0.894
2400images_0101_10lbins_10cbins	0.926	0.995	0.855	0.920
2400images_0101_10lbins_500cbins	0.968	0.993	0.942	0.967
2400images_0101_10lbins_50cbins	0.903	0.986	0.818	0.895
2400images_0101_500lbins_1000cbins	0.928	1.0	0.856	0.922
2400images_0101_500lbins_100cbins	0.888	1.0	0.777	0.875
2400images_0101_500lbins_10cbins	0.893	1.0	0.787	0.881
2400images_0101_500lbins_500cbins	0.916	1.0	0.832	0.908
2400images_0101_500lbins_50cbins	0.889	1.0	0.778	0.875
2400images_0101_50lbins_1000cbins	0.965	0.996	0.934	0.964
2400images_0101_50lbins_100cbins	0.883	0.993	0.773	0.869
2400images_0101_50lbins_10cbins	0.901	0.995	0.805	0.890
2400images_0101_50lbins_500cbins	0.963	0.997	0.928	0.961
2400images_0101_50lbins_50cbins	0.881	0.994	0.767	0.865
2400images_0110_1000lbins	0.897	0.999	0.795	0.886
2400images_0110_100lbins	0.895	0.996	0.793	0.883
2400images_0110_10lbins	0.937	0.994	0.88	0.933
2400images_0110_500lbins	0.891	1.0	0.782	0.877
2400images_0110_50lbins	0.901	0.994	0.806	0.890
2400images_0111_1000lbins_1000cbins	0.928	1.0	0.856	0.922
2400images_0111_1000lbins_100cbins	0.899	0.999	0.799	0.888
2400images_0111_1000lbins_10cbins	0.904	0.999	0.808	0.894
2400images_0111_1000lbins_500cbins	0.920	1.0	0.840	0.913
2400images_0111_1000lbins_50cbins	0.898	0.999	0.798	0.887
2400images_0111_100lbins_1000cbins	0.963	0.997	0.928	0.961
2400images_0111_100lbins_100cbins	0.903	0.997	0.809	0.893
2400images_0111_100lbins_10cbins	0.915	0.998	0.831	0.907
2400images_0111_100lbins_500cbins	0.96	0.999	0.920	0.958
2400images_0111_100lbins_50cbins	0.901	0.997	0.805	0.891
2400images_0111_10lbins_1000cbins	0.964	0.992	0.935	0.963
2400images_0111_10lbins_100cbins	0.951	0.992	0.909	0.949
2400images_0111_10lbins_10cbins	0.960	0.998	0.922	0.958

Continued on next page

Table 1 – *Continued from previous page*

Experiment	Accuracy	Precision	Recall	F-score
2400images_0111_10lbins_500cbins	0.966	0.991	0.941	0.965
2400images_0111_10lbins_50cbins	0.951	0.990	0.911	0.949
2400images_0111_500lbins_1000cbins	0.938	1.0	0.877	0.934
2400images_0111_500lbins_100cbins	0.894	1.0	0.789	0.882
2400images_0111_500lbins_10cbins	0.898	1.0	0.797	0.887
2400images_0111_500lbins_500cbins	0.923	1.0	0.847	0.917
2400images_0111_500lbins_50cbins	0.895	1.0	0.790	0.882
2400images_0111_50lbins_1000cbins	0.965	0.996	0.934	0.964
2400images_0111_50lbins_100cbins	0.913	0.996	0.83	0.905
2400images_0111_50lbins_10cbins	0.929	0.997	0.861	0.924
2400images_0111_50lbins_500cbins	0.965	0.997	0.932	0.963
2400images_0111_50lbins_50cbins	0.915	0.997	0.832	0.907
2400images_1000	0.942	0.946	0.937	0.941
2400images_1001_1000cbins	0.960	0.989	0.931	0.959
2400images_1001_100cbins	0.956	0.974	0.937	0.955
2400images_1001_10cbins	0.962	0.974	0.948	0.961
2400images_1001_500cbins	0.963	0.989	0.937	0.962
2400images_1001_50cbins	0.951	0.958	0.944	0.951
2400images_1010	0.954	0.963	0.945	0.954
2400images_1011_1000cbins	0.956	0.989	0.923	0.955
2400images_1011_100cbins	0.965	0.983	0.947	0.965
2400images_1011_10cbins	0.967	0.981	0.953	0.967
2400images_1011_500cbins	0.96	0.987	0.932	0.958
2400images_1011_50cbins	0.963	0.968	0.958	0.963
2400images_1100_1000lbins	0.901	0.999	0.804	0.891
2400images_1100_100lbins	0.915	0.994	0.835	0.908
2400images_1100_10lbins	0.956	0.973	0.937	0.955
2400images_1100_500lbins	0.901	1.0	0.802	0.890
2400images_1100_50lbins	0.927	0.992	0.861	0.922
2400images_1101_1000lbins_1000cbins	0.935	1.0	0.870	0.930
2400images_1101_1000lbins_100cbins	0.905	0.999	0.811	0.895
2400images_1101_1000lbins_10cbins	0.909	0.999	0.819	0.900
2400images_1101_1000lbins_500cbins	0.927	1.0	0.854	0.921
2400images_1101_1000lbins_50cbins	0.904	0.999	0.809	0.894
2400images_1101_100lbins_1000cbins	0.974	0.999	0.95	0.973
2400images_1101_100lbins_100cbins	0.921	0.996	0.845	0.914
2400images_1101_100lbins_10cbins	0.933	0.997	0.869	0.928
2400images_1101_100lbins_500cbins	0.969	0.999	0.939	0.968
2400images_1101_100lbins_50cbins	0.921	0.997	0.845	0.914

Continued on next page

Table 1 – *Continued from previous page*

Experiment	Accuracy	Precision	Recall	F-score
2400images_1101_10lbins_1000cbins	0.974	0.995	0.952	0.973
2400images_1101_10lbins_100cbins	0.964	0.989	0.939	0.963
2400images_1101_10lbins_10cbins	0.971	0.995	0.947	0.970
2400images_1101_10lbins_500cbins	0.975	0.993	0.956	0.974
2400images_1101_10lbins_50cbins	0.963	0.986	0.939	0.962
2400images_1101_500lbins_1000cbins	0.947	1.0	0.894	0.944
2400images_1101_500lbins_100cbins	0.903	1.0	0.806	0.892
2400images_1101_500lbins_10cbins	0.907	1.0	0.815	0.898
2400images_1101_500lbins_500cbins	0.930	1.0	0.860	0.924
2400images_1101_500lbins_50cbins	0.902	1.0	0.805	0.892
2400images_1101_50lbins_1000cbins	0.975	0.996	0.953	0.974
2400images_1101_50lbins_100cbins	0.936	0.993	0.877	0.932
2400images_1101_50lbins_10cbins	0.951	0.996	0.905	0.948
2400images_1101_50lbins_500cbins	0.975	0.997	0.952	0.974
2400images_1101_50lbins_50cbins	0.936	0.996	0.876	0.932
2400images_1110_1000lbins	0.908	0.999	0.817	0.899
2400images_1110_100lbins	0.930	0.997	0.862	0.925
2400images_1110_10lbins	0.973	0.987	0.957	0.972
2400images_1110_500lbins	0.907	1.0	0.815	0.898
2400images_1110_50lbins	0.952	0.995	0.907	0.949
2400images_1111_1000lbins_1000cbins	0.941	1.0	0.882	0.937
2400images_1111_1000lbins_100cbins	0.910	0.999	0.822	0.902
2400images_1111_1000lbins_10cbins	0.916	1.0	0.832	0.908
2400images_1111_1000lbins_500cbins	0.930	1.0	0.860	0.925
2400images_1111_1000lbins_50cbins	0.911	0.999	0.823	0.902
2400images_1111_100lbins_1000cbins	0.973	0.998	0.948	0.972
2400images_1111_100lbins_100cbins	0.936	0.998	0.875	0.932
2400images_1111_100lbins_10cbins	0.948	0.998	0.897	0.945
2400images_1111_100lbins_500cbins	0.972	0.999	0.946	0.971
2400images_1111_100lbins_50cbins	0.937	0.998	0.877	0.933
2400images_1111_10lbins_1000cbins	0.972	0.994	0.950	0.972
2400images_1111_10lbins_100cbins	0.976	0.993	0.959	0.976
2400images_1111_10lbins_10cbins	0.980	0.997	0.963	0.980
2400images_1111_10lbins_500cbins	0.973	0.993	0.954	0.973
2400images_1111_10lbins_50cbins	0.976	0.991	0.961	0.976
2400images_1111_500lbins_1000cbins	0.951	1.0	0.902	0.948
2400images_1111_500lbins_100cbins	0.911	1.0	0.822	0.902
2400images_1111_500lbins_10cbins	0.917	1.0	0.835	0.910
2400images_1111_500lbins_500cbins	0.936	1.0	0.873	0.932

Continued on next page

Table 1 – *Continued from previous page*

Experiment	**Accuracy**	**Precision**	**Recall**	**F-score**
2400images_1111_500lbins_50cbins	0.910	1.0	0.821	0.902
2400images_1111_50lbins_1000cbins	0.974	0.996	0.952	0.973
2400images_1111_50lbins_100cbins	0.959	0.996	0.922	0.958
2400images_1111_50lbins_10cbins	0.966	0.998	0.934	0.965
2400images_1111_50lbins_500cbins	0.976	0.997	0.955	0.976
2400images_1111_50lbins_50cbins	0.961	0.997	0.924	0.959

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia
2. Dudly Knox Library
   Naval Postgraduate School
   Monterey, California